Canonical Polyadic Decomposition based on joint eigenvalue decomposition

Xavier Luciani, Laurent Albera

Canonical Polyadic Decomposition based on joint eigenvalue decomposition

Xavier Luciani[a,b], Laurent Albera[c,d,e,*]

[a]*Aix Marseille Université, CNRS, ENSAM, LSIS, UMR 7296, Marseille, F-13397, France.*
[b]*Université de Toulon, CNRS, LSIS, UMR 7296, La Garde, F-83957, France.*
[c]*Inserm, UMR 642, Rennes, F-35000, France*
[d]*LTSI, University of Rennes 1, Rennes, F-35000, France*
[e]*Inria, Centre Inria Rennes - Bretagne Atlantique, Rennes, F-35000, France*

Abstract

A direct algorithm based on Joint EigenValue Decomposition (JEVD) has been proposed to compute the Canonical Polyadic Decomposition (CPD) of multi-way arrays (tensors). The iterative part of our method is thus limited to the JEVD computation. At this occasion we also propose an original JEVD technique. Most of the iterative CPD algorithms such as ALS have been shown by means of practical studies to suffer from convergence problems (local minima, slow convergence or high computational cost per iteration). On the other hand, direct methods seem in practice to confine these disadvantages but impose some restrictive necessary conditions. In this context, our proposed algorithm involves less restrictive necessary conditions than other recent direct approaches and a limited computational complexity. It has been compared to reference (direct and non-direct) algorithms on synthetic arrays and real spectroscopic data. These numerical examples highlight the main advantages of the proposed methods to solve both the JEVD and CPD problems.

Keywords: multi-way arrays, direct canonical polyadic decomposition, PARAFAC, joint eigenvalue decomposition, fluorescence, over-factoring

1. Introduction

In this paper, we mainly propose a direct algorithm for the canonical polyadic decomposition of real or complex-valued tensors (assimilated to multi-way arrays) using the Joint EigenValue Decomposition (JEVD) of a set of non-defective matrices. The present contribution is actually twofold since we jointly propose an algorithm to solve the JEVD problem. Tensor decomposition plays a wider and wider role in numerous application areas such as Psychometric [1], Signal Processing for Biomedical Engineering [2, 3, 4], Sensor array [5, 6, 7], Arithmetic Complexity [8] and Chemometrics [9, 10]. Thanks to its uniqueness properties [11, 12, 13, 14, 15, 16],

*Corresponding author. Address: University of Rennes 1, LTSI, Rennes F-35000, France. Phone: +33 2 23 23 50 58, fax: +33 2 23 23 69 17

Email addresses: luciani@univ-tln.fr (Xavier Luciani), laurent.albera@univ-rennes1.fr, http://perso.univ-rennes1.fr/laurent.albera/ (Laurent Albera)

the polyadic decomposition introduced in 1927 by Hitchcock [17] is probably the most popular nowadays. In fact, it is now best known as CANonical DECOMPosition (CANDECOMP) [1], PARAllel FACtor analysis (PARAFAC) [18] or CANDECOMP/PARAFAC (CP). In order to be consistent and honor the original work we will keep the acronym CPD, which stands for Canonical Polyadic Decomposition.

More precisely, a polyadic decomposition of an array is a sum of rank-one terms that yields an exact fit [17]. The CPD is then defined as the minimal polyadic decomposition. The rank of an array may be thus defined as the minimal number of rank-1 tensors needed to achieve the CPD.

Many algorithms have been proposed in order to compute the CPD of multi-way arrays. One of the most famous algorithms, due to its speed and ease of implementation, resorts to an iterative Alternating Least Squares (ALS) procedure [18]. Other iterative algorithms based on first and second order optimization methods such as gradient, Gauss-Newton, Levenberg-Marquardt or conjugate gradient have also been proposed (see [19] [20, 21, 22] for a full comparison). Recently, a set of iterative algorithms based on a reduced functional has been introduced in [23].These last algorithms bring qualitative information on the solution but the counter part is a longer computational time. Furthermore, an Enhanced Line Search (ELS) procedure has been proposed in [24] in order to speed up the ALS algorithm. ELS extension to other iterative CPD algorithm and efficiency of the ALS-ELS algorithm has been highlighted in [21]. However, in spite of this refinement, the ALS algorithm suffers from a classical drawback. Indeed, nothing ensures its global convergence and it can be stuck in local minima. More generaly, iterative approaches show convergence problems when several factors of the CPD are correlated.

In the meantime, a few direct approaches have been proposed. One can mention the DTLD approach [25]. However it is restricted to three-way arrays and provide poor results [26, 20]. Thereby this kind of solution is generally used as a way of initializing iterative methods. Other direct approaches have been proposed in the literature as well but not yet compared numerically in studies such as the ones mentioned above. These methods rephrase the CPD as the simultaneous diagonalization, by equivalence [27, 28, 29] or congruence [15], of a set of matrices. The CPD problem can also be translated into a simultaneous generalized Schur decomposition, with orthogonal unknowns, as shown in [29]. Direct methods compute the CPD by solving an alternative algebra problem of lower dimensions but they do not provide a solution in terms of least squares contrarily to the ALS and derivative-based techniques. The reformulated problem is usually solved by means of a Jacobi-like procedure.

We thus propose here a new formulation of the CPD as a JEVD problem leading to a novel direct solution, named DIAG (DIrect AlGorithm for canonical polyadic decomposition), involving less restrictive necessary conditions than the "Closed Form Solution" (CFS) presented in [27, 28]. Recall that the CFS algorithm requires that the rank of the considered CPD array does not exceed two of the dimensions of the array. At this occasion we also propose an original Jacobi-like JEVD algorithm, called JDTM (Joint Diagonalization algorithm based on Targeting hyperbolic Matrices). Numerical examples highlight the main advantages of the proposed methods to solve the JEVD and CPD problems. Note that the DIAG method can be seen as a generalization of the BIOME approach [30] to the case of unsymmetric arrays. JDTM and DIAG have been presented briefly in two separate conference papers [31, 32], respectively. In [32] DIAG was associated to another JEVD algorithm and was called SALT (SemiALgebraic Tensor decomposition). The present paper details theoretical aspects of both algorithms in sections 2 and 3, respectively including their extension to the complex case which is not trivial and

their computational complexity. In addition subsection 3.5 is dedicated to the comparison of necessary conditions of different CPD algorithms, namely ALS, CFS and DIAG . Numerical results are also emphasized in section 4 which illustrate the main features of the DIAG approach, notably the problem of over-factoring is addressed. Finally a concrete application to fluorescence spectroscopy is proposed in section 5.

2. Joint eigenvalue decomposition of non-defective matrices

We use the following consistent notations in the whole paper: vectors, matrices and tensors are denoted by lower case boldface ($\boldsymbol{a}$), upper case boldface ($\boldsymbol{A}$) and upper case boldface calligraphic ($\boldsymbol{\mathcal{A}}$) letters respectively. The i-th entry of vector $\boldsymbol{a}$ is denoted by a_i while A_{ij} is the (i, j)-th component of matrix $\boldsymbol{A}$. Entry $(i_1, \ldots, i_Q)$ of any Q-order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_Q}$ or $\mathbb{C}^{I_1 \times \cdots \times I_Q}$ ($Q > 2$) is denoted by $\mathcal{T}_{i_1,\cdots,i_Q}$. Outer product, Kronecker product and Khatri-Rao product are denoted by $\circ$, $\otimes$ and $\odot$, respectively. Moore-Penrose matrix inverse, euclidean and frobenius norm are denoted by $\sharp$, $\|\boldsymbol{E}^{(k)}\|_F$ and $\|.\|_F$, respectively. We define $[x; y]_{\mathbb{N}} = [x; y] \cap \mathbb{N}$. $\lfloor . \rfloor$ denotes the floor function. Complex modulus and conjugate of any complex z are denoted by $|z|$ and $\overline{z}$ respectively. The imaginary unit is denoted by i.
Givens and hyperbolic rotation matrices are denoted by $\boldsymbol{G}$ and $\boldsymbol{H}$, respectively. For instance in the real case, $\boldsymbol{G}(\theta_{ij})$ and $\boldsymbol{H}(\phi_{ij})$ are equal to the identity matrix, at the exception of the elements:

$$G(\theta_{ij})_{ii} = G(\theta_{ij})_{jj} = \cos(\theta_{ij}) \qquad G(\theta_{ij})_{ij} = -G(\theta_{ij})_{ji} = \sin(\theta_{ij})$$
$$H(\phi_{ij})_{ii} = H(\phi_{ij})_{jj} = \cosh(\phi_{ij}) \qquad H(\phi_{ij})_{ij} = H(\phi_{ij})_{ji} = \sinh(\phi_{ij})$$

The JEVD problem consists in finding an eigenvector matrix $\boldsymbol{A}$ from a set of non-defective matrices $\boldsymbol{M}^{(k)}$ satisfying:

$$\forall k \in [1; K]_{\mathbb{N}},\ \boldsymbol{M}^{(k)} = \boldsymbol{A}\boldsymbol{D}^{(k)}\boldsymbol{A}^{-1}, \tag{1}$$

where the K diagonal matrices $\boldsymbol{D}^{(k)}$ are unknown. One could solve these EVDs separately, and retain the solution that leads to the best estimate regarding the considered application. However, as explained in [29], it is safer from a numerical point of view to decompose the K matrices $\boldsymbol{M}^{(k)}$ simultaneously, in some optimal sense, especially when the perturbation of these matrices may have caused eigenvalues to cross each other. Indeed, in practice only noisy observations of the K matrices $\boldsymbol{M}^{(k)}$ are clustered and it is well known that, when eigenvalues are close, the eigenvectors in a single EVD may be strongly affected by small perturbations [33]. The reason is that for coinciding eigenvalues only the corresponding eigenspace is defined; different directions in this subspace will emerge as eigenvectors for different infinitesimal perturbations. When this happens for one or more of the matrices in the JEVD problem, the other matrices may still allow to identify the actual eigenvectors. This follows theorem proved in [29]:

Theorem 1. *The JEVD is unique up to a permutation and a scaling of the columns of $\boldsymbol{A}$ if and only if all the columns of the $K \times N$ matrix $\boldsymbol{E}$, whose (k, n)-th component $E_{k,n}$ is equal to $D^{(k)}_{n,n}$, are not proportional.*

Note that in order to ensure uniqueness of the JEVD up to permutation and scale indeterminacies, we will assume in the sequel that the K involved diagonal matrices $\boldsymbol{D}^{(k)}$ fulfil the condition given in Theorem 1.

Few papers have proposed numerical solutions to the JEVD problem. All of them adapted Jacobi's principle to the search for a non-singular and non-necessarily orthogonal eigenmatrix $\boldsymbol{A}$

by using a suitable factorization, which is not reduced to the product of Givens matrices. This domination of Jacobi-like methods is due to their good convergence properties [34].

Two main kinds of Jacobi-like algorithms have been developed in this context, based on different matrix factorizations. Originally, several authors had recourse to the QR factorization of $\boldsymbol{A}$ in order to compute the different sets of eigenvalues [35, 36]. Arguing that these QR-algorithms suffer from convergence problems, Fu and Gao proposed an effective sh-rt algorithm [37] based on the polar decomposition. Indeed the polar decomposition has been used favourably for eigenvalue decomposition purpose since a long time [38, 39, 34] and also for joint diagonalization by congruence [40]. Then the JUST algorithm was introduced in [41] as a variation of the sh-rt approach for which the iterative computation of the hyperbolic matrix is made by minimizing an alternative criterion. We propose here a third criterion and an appropriate optimization method, giving birth to the JDTM algorithm. Another JEVD approach based on LU factorization and called JET was introduced in [32] for real-valued matrices.

The real case is addressed in the three following subsections. The extension to the complex case is described in subsection 2.4. JDTM algorithm has been compared to JUST and sh-rt algorithms in various situations involving real matrices. Significant numerical results are given in section 4.1.

2.1. A Jacobi-like process

In this subsection, all matrices are square matrices of order N. Polar matrix decomposition states that any non-singular real matrix can be factorized into the product of an orthogonal matrix $\boldsymbol{Q}$ and a symmetric positive semidefinite matrix $\boldsymbol{S}$. It is well known that $\boldsymbol{Q}$ can be decomposed into a product of Givens rotation matrices $\boldsymbol{G}(\theta_{ij})$ and a unitary diagonal matrix. In the same way, it has been shown that $\boldsymbol{S}$ can be decomposed into a product of hyperbolic rotation matrices $\boldsymbol{H}(\phi_{ij})$ and diagonal matrices [40]. Thereby, due to the indeterminacies of the JEVD problem mentioned in theorem 1 and taking into account that diagonal, hyperbolic and Givens matrices commute, the matrix $\boldsymbol{A}$ solving the JEVD problem given by (1) can be chosen as a product of Givens and hyperbolic rotation matrices:

$$\boldsymbol{A} = \prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \boldsymbol{G}(\theta_{ij})\boldsymbol{H}(\phi_{ij}). \tag{2}$$

Inserting (2) into (1) and using the fact that $\boldsymbol{H}(\phi_{ij})^{-1} = \boldsymbol{H}(-\phi_{ij})$ we get:

$$\forall k \in [1;K]_{\mathbb{N}},\ \boldsymbol{D}^{(k)} = \left(\prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \boldsymbol{G}(\theta_{ij})^{\mathsf{T}}\boldsymbol{H}(-\phi_{ij})\right) \boldsymbol{M}^{(k)} \left(\prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \boldsymbol{G}(\theta_{ij})\boldsymbol{H}(\phi_{ij})\right), \tag{3}$$

but we prefer the simpler formulation:

$$\forall k \in [1;K]_{\mathbb{N}},\ \boldsymbol{D}^{(k)} = \left(\prod_{m=1}^{M} \boldsymbol{H}(-\phi_m)\boldsymbol{G}(\theta_m)^{\mathsf{T}}\right) \boldsymbol{M}^{(k)} \left(\prod_{m=1}^{M} \boldsymbol{G}(\theta_m)\boldsymbol{H}(\phi_m)\right), \tag{4}$$

where each integer m of $[1;M]_{\mathbb{N}}$ stands for a couple (i,j) with $1 \leq i < j \leq N$. It is worth mentioning that any Givens or hyperbolic matrix is defined by only one parameter (angle). Therefore, ideally we have to find a set of $M = N(N-1)/2$ couples of parameters $\{(\theta_{ij}, \phi_{ij})\}_{1\leq i<j\leq N}$ in order to get (1). Instead of simultaneously identifying these M couples of parameters, a Jacobi-like

procedure will repeat sequences of $2M$ successive optimizations until convergence. Each optimization is performed with respect to only one parameter. A sequence of $2M$ optimizations is generally called a sweep. As a result, $N_s M$ couples of Givens and hyperbolic matrices are used in practice to identify $\boldsymbol{A}$, where N_s is the number of sweeps. We thus look for a matrix $\boldsymbol{A}$ of the form $\boldsymbol{A} = \prod_{n_s=1}^{N_s} \prod_{m=1}^{M} \boldsymbol{G}(\theta_m^{n_s})\boldsymbol{H}(\phi_m^{n_s})$. The idea is to iteratively diagonalize the $\boldsymbol{M}^{(k)}$ matrices by sequentially optimizing with respect to $\theta_m^{n_s}$ and $\phi_m^{n_s}$ for each value of m and n_s. Hence the first sweep ($n_s = 1$) consists on the following transformations:

$$\forall k \in [1;K]_{\mathbb{N}},\ \boldsymbol{N}^{(k,1,1)} = \boldsymbol{G}(\theta_1^1)^{\top}\boldsymbol{M}^{(k)}\boldsymbol{G}(\theta_1^1), \tag{5}$$

$$\forall (k,m) \in [1;K]_{\mathbb{N}} \times [1;M]_{\mathbb{N}},\ \boldsymbol{M}^{(k,m,1)} = \boldsymbol{H}(-\phi_m^1)\boldsymbol{N}^{(k,m,1)}\boldsymbol{H}(\phi_m^1). \tag{6}$$

$$\forall (k,m) \in [1;K]_{\mathbb{N}} \times [2;M]_{\mathbb{N}}\ \boldsymbol{N}^{(k,m,1)} = \boldsymbol{G}(\theta_m^1)^{\top}\boldsymbol{M}^{(k,m-1,1)}\boldsymbol{G}(\theta_m^1) \tag{7}$$

Then the following sweeps ($1 < n_s \leq N_s$) follow the same scheme:

$$\forall (k,n_s) \in [1;K]_{\mathbb{N}} \times [2;N_s]_{\mathbb{N}},\ \boldsymbol{N}^{(k,1,n_s)} = \boldsymbol{G}(\theta_1^{n_s})^{\top}\boldsymbol{M}^{(k,M,n_s-1)}\boldsymbol{G}(\theta_1^{n_s}), \tag{8}$$

$$\forall (k,m,n_s) \in [1;K]_{\mathbb{N}} \times \in [1;M]_{\mathbb{N}} \times [2;N_s]_{\mathbb{N}},\ \boldsymbol{M}^{(k,m,n_s)} = \boldsymbol{H}(-\phi_m^{n_s})\boldsymbol{N}^{(k,m,n_s)}\boldsymbol{H}(\phi_m^{n_s}). \tag{9}$$

$$\forall (k,m,n_s) \in [1;K]_{\mathbb{N}} \times \in [2;M]_{\mathbb{N}} \times [2;N_s]_{\mathbb{N}},\ \boldsymbol{N}^{(k,m,n_s)} = \boldsymbol{G}(\theta_m^{n_s})^{\top}\boldsymbol{M}^{(k,m-1,n_s)}\boldsymbol{G}(\theta_m^{n_s}), \tag{10}$$

Thereby, the optimal corresponding Givens and hyperbolic matrices are sequentially computed in order to get K diagonal matrices $\boldsymbol{M}^{(k,M,N_s)}$ at the end of the process.

2.2. Optimization of matrix angles

A natural criterion to compute the optimal (m, n_s)-th Givens angle $\theta_m^{n_s}$ is thus to minimize the sum of the euclidean norms of the off-diagonal terms of the K matrices $\boldsymbol{N}^{(k,m,n_s)}$:

$$\zeta_G(\theta_m^{n_s}) = \sum_{k=1}^{K} \sum_{\substack{p=1,q=1 \\ p\neq q}}^{N,N} \left(N_{pq}^{(k,m,n_s)}\right)^2. \tag{11}$$

This criterion is the generalization of the original Jacobi criterion to the joint diagonalization context. Since Givens matrices are orthogonal, the same definition of $\boldsymbol{N}^{(k,m,n_s)}$ holds in both the joint diagonalization by congruence and JEVD cases and thus the same optimization algorithms can be used. For instance, our proposed algorithm resorts to the same approach as the JAD algorithm described in [42] whereas the sh-rt and JUST algorithms use their own minimization scheme.

Once the optimal Givens matrix $\boldsymbol{G}(\theta_m^{n_s})$ is computed, different criteria can be used for the optimal computation of $\boldsymbol{H}(\phi_m^{n_s})$. This is the main difference between the three JEVD algorithms. The sh-rt method aims at minimizing the Frobenius norm of $\boldsymbol{M}^{(h,m,n_s)}$ where h is found such that $\left|M_{ii}^{(h,m,n_s)} - M_{jj}^{(h,m,n_s)}\right| = \max_{1\leq k\leq K} \left|M_{ii}^{(k,m,n_s)} - M_{jj}^{(k,m,n_s)}\right|$, whereas the JUST algorithm resorts to criterion (11) by replacing $\boldsymbol{N}^{(k,m,n_s)}$ by $\boldsymbol{M}^{(k,m,n_s)}$. Instead of minimizing all the (off-diagonal) entries, we propose to target two particular off-diagonal entries of $\boldsymbol{M}^{(k,m,n_s)}$: if m corresponds to the $(i,j)_{i<j}$ couple, we simply aim at computing the optimal $M_{ij}^{(k,m,n_s)}$ and $M_{ji}^{(k,m,n_s)}$ components by using a "targeting" hyperbolic matrix. It is noteworthy that the transformation (9) affects the i-th and j-th rows and the i-th and j-th columns of $\boldsymbol{M}^{k,m,n_s}$ but only the (i,j) and the (j,i) components are twice affected by the hyperbolic matrix and its inverse. Hence our choice to focus on the latter.

Therefore, our Joint Diagonalization algorithm based on Targeting hyperbolic Matrices (JDTM) resorts to the following alternative criterion ζ_H^{JDTM} for the computation of the hyperbolic matrix:

$$\zeta_H^{JDTM}(\phi_m^{n_s}) = \sum_{k=1}^{K} \left(M_{ij}^{(k,m,n_s)}\right)^2 + \left(M_{ji}^{(k,m,n_s)}\right)^2, \tag{12}$$

Targeting some components was originally proposed by Souloumiac in a different context [40]. In the case of Givens matrices we showed that the optimizations of criteria (11) and (12) were mathematically equivalent.

Now, let us look at the components of $\boldsymbol{M}^{(k,m,n_s)}$. As previously mentioned, we only consider the (i, j)-th and (j, i)-th components which are given by:

$$M_{ij}^{(k,m,n_s)} = \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \frac{\sinh(2\phi_m^{n_s})}{2} + N_{ij}^{(k,m,n_s)} \cosh(\phi_m^{n_s})^2 - N_{ji}^{(k,m,n_s)} \sinh(\phi_m^{n_s})^2, \tag{13}$$

$$M_{ji}^{(k,m,n_s)} = \left(N_{jj}^{(k,m,n_s)} - N_{ii}^{(k,m,n_s)}\right) \frac{\sinh(2\phi_m^{n_s})}{2} - N_{ij}^{(k,m,n_s)} \sinh(\phi_m^{n_s})^2 + N_{ji}^{(k,m,n_s)} \cosh(\phi_m^{n_s})^2. \tag{14}$$

Furthermore we can write that:

$$\left(M_{ij}^{(k,m,n_s)}\right)^2 + \left(M_{ji}^{(k,m,n_s)}\right)^2 = \frac{\left(M_{ij}^{(k,m,n_s)} + M_{ji}^{(k,m,n_s)}\right)^2}{2} + \frac{\left(M_{ij}^{(k,m,n_s)} - M_{ji}^{(k,m,n_s)}\right)^2}{2}. \tag{15}$$

The first term of the right-hand side does not depend on $\phi_m^{n_s}$. Indeed, we derive from (13) and (14) the following equality:

$$\frac{\left(M_{ij}^{(k,m,n_s)} + M_{ji}^{(k,m,n_s)}\right)^2}{2} = \frac{\left(N_{ij}^{(k,m,n_s)} + N_{ji}^{(k,m,n_s)}\right)^2}{2}. \tag{16}$$

Thereby minimizing ζ_H^{JTDM} is equivalent to minimize the λ function defined by:

$$\lambda(\phi_m^{n_s}) = \sum_{k=1}^{K} \left(M_{ij}^{(k,m,n_s)} - M_{ji}^{(k,m,n_s)}\right)^2. \tag{17}$$

We denote by $\boldsymbol{y}^{(m,n_s)}$ the column vector of $\mathbb{R}^K$ defined by $y_k^{(m,n_s)} = M_{ij}^{(k,m,n_s)} - M_{ji}^{(k,m,n_s)}$, so that $\lambda(\phi_m^{n_s}) = \boldsymbol{y}^{(m,n_s)\top}\boldsymbol{y}^{(m,n_s)}$. It is easily shown that the system of linear equations (13) and (14) can be rewritten such that:

$$\boldsymbol{y}^{(m,n_s)} = \boldsymbol{W}^{(m,n_s)}\boldsymbol{x}(\phi_m^{n_s}), \tag{18}$$

with:

$$\boldsymbol{W}^{(m,n_s)} = \begin{bmatrix} N_{ii}^{(1,m,n_s)} - N_{jj}^{(1,m,n_s)} & N_{ij}^{(1,m,n_s)} - N_{ji}^{(1,m,n_s)} \\ \vdots & \vdots \\ N_{ii}^{(K,m,n_s)} - N_{jj}^{(K,m,n_s)} & N_{ij}^{(K,m,n_s)} - N_{ji}^{(K,m,n_s)} \end{bmatrix} \quad ; \quad \boldsymbol{x}(\phi_m^{n_s}) = \begin{bmatrix} \sinh(2\phi_m^{n_s}) \\ \cosh(2\phi_m^{n_s}) \end{bmatrix}.$$

Now defining the diagonal 2×2 matrix $\boldsymbol{J}$ such that $J_{11} = -J_{22} = -1$ and observing that $\boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{J} \boldsymbol{x}(\phi_m^{n_s}) = 1$, we have thus to minimize the quantity $\boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} \boldsymbol{x}(\phi_m^{n_s})$ under

the constraint that $\boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{J}\boldsymbol{x}(\phi_m^{n_s}) = 1$. This can be done using the Lagrange multipliers strategy. Thereby, we have to minimize the L function given by:

$$L(\boldsymbol{x}(\phi_m^{n_s}), \mu(\phi_m^{n_s})) = \boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} \boldsymbol{x}(\phi_m^{n_s}) - \mu(\phi_m^{n_s}) \boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{J}\boldsymbol{x}(\phi_m^{n_s}). \tag{19}$$

Differentiation with respect to $\boldsymbol{x}(\phi_m^{n_s})$ leads to:

$$\boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} \boldsymbol{x}(\phi_m^{n_s}) = \mu(\phi_m^{n_s}) \boldsymbol{J}\boldsymbol{x}(\phi_m^{n_s}). \tag{20}$$

Since $\boldsymbol{J}^{-1} = \boldsymbol{J}$ we have:

$$\boldsymbol{J}\boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} \boldsymbol{x}(\phi_m^{n_s}) = \mu(\phi_m^{n_s}) \boldsymbol{x}(\phi_m^{n_s}). \tag{21}$$

Thus, $\mu(\phi_m^{n_s})$ and $\boldsymbol{x}(\phi_m^{n_s})$ are associated eigenvalue and eigenvector of matrix $\boldsymbol{J}\boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)}$. More particularly, we have the following lemma:

Lemma 1. *If the columns of $\boldsymbol{W}^{(m,n_s)}$ are different then $\boldsymbol{J}\boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)}$ has two nonzero eigenvalues of opposite sign and $\boldsymbol{x}(\phi_m^{n_s})$ is the eigenvector associated to the positive eigenvalue.*

Proof 1. *Let $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ be the column vectors of matrix $\boldsymbol{W}^{(m,n_s)}$. Both belong to $\mathbb{R}^K$, equipped with the Euclidean norm and we define $a = \boldsymbol{w}_1{}^\top \boldsymbol{w}_1$, $b = \boldsymbol{w}_1{}^\top \boldsymbol{w}_2$ and $c = \boldsymbol{w}_2{}^\top \boldsymbol{w}_2$. Hence a, b and c denote the squared euclidean norm of $\boldsymbol{w}_1$, the scalar product between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ and the squared Euclidean norm of $\boldsymbol{w}_2$ respectively. Hence,*

$$\boldsymbol{J}\boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} = \begin{bmatrix} -a & -b \\ b & c \end{bmatrix}$$

The characteristic polynomial is then:

$$P(\alpha) = \alpha^2 + (a - c)\alpha + (b^2 - ca) \tag{22}$$

and the discriminant is:

$$\begin{aligned} \Delta &= (a-c)^2 - 4b^2 + 4ca \\ &= (a + c - 2b)(a + c + 2b) \\ &= \| \boldsymbol{w}_1 - \boldsymbol{w}_2 \|^2 \| \boldsymbol{w}_1 + \boldsymbol{w}_2 \|^2 \end{aligned}$$

Thereby, since $\boldsymbol{w}_1 \neq \boldsymbol{w}_2$, $\Delta > 0$ and $\boldsymbol{J}\boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)}$ is diagonalizable and admits two distinct eigenvalues α_1 and α_2. Then we have:

$$\begin{aligned} \alpha_1 \alpha_2 &= \frac{(a-c)^2 - \Delta}{4a^2} \\ &= \frac{b^2 - ac}{a^2} \end{aligned}$$

The Cauchy-Schwartz inequality gives $b^2 < ac$ hence $\alpha_1 \alpha_2 < 0$.
We now demonstrate the second part of the lemma. Multiplying (21) by $\boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{J}$ yields:

$$\begin{aligned} \boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} \boldsymbol{x}(\phi_m^{n_s}) &= \mu(\phi_m^{n_s}) \boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{J}\boldsymbol{x}(\phi_m^{n_s}), \\ &= \mu(\phi_m^{n_s}). \end{aligned} \tag{23}$$

The quadratic form $\boldsymbol{x}(\phi_m^{n_s})^\top \boldsymbol{W}^{(m,n_s)\top} \boldsymbol{W}^{(m,n_s)} \boldsymbol{x}(\phi_m^{n_s})$ is positive thus $\mu(\phi_m^{n_s})$ is positive too.

Hence the previous lemma allows us to easily compute $\boldsymbol{x}(\phi_m^{n_s})$ from $\boldsymbol{W}^{(m,n_s)}$ and $\phi_m^{n_s}$ is deduced from the definition of $\boldsymbol{x}(\phi_m^{n_s})$:

$$\phi_m^{n_s} = \frac{1}{2}\text{atanh}\left(\frac{x(\phi_m^{n_s})_1}{x(\phi_m^{n_s})_2}\right). \tag{24}$$

Algorithm 1 summarizes the proposed method.

Algorithm 1: Summary of the JDTM algorithm

1: Define a threshold ε and a maximal number of sweep N_s^{max}
2: Initialize $\boldsymbol{A}$ with the identity matrix;
3: $n_s = 1$;
4: **while** $\sum_k \sum_{p\neq q}(M_{p,q}^{(k)})^2 > \varepsilon$ and $n_s \leq N_s^{max}$ **do**
5: $m = 1$;
6: **for** $i = 1$ to $N-1$ **do**
7: **for** $j = i+1$ to N **do**
8: Compute the optimal angle $\theta_m^{n_s}$ corresponding to the couple (i, j) and build $\boldsymbol{G}(\theta_m^{n_s})$;
9: Replace the K matrices $\boldsymbol{M}^{(k)}$ by $\boldsymbol{G}(\theta_m^{n_s})^\top \boldsymbol{M}^{(k)} \boldsymbol{G}(\theta_m^{n_s})$;
10: Compute the optimal angle $\phi_m^{n_s}$ corresponding to the couple (i, j) and build $\boldsymbol{H}(\phi_m^{n_s})$;
11: Replace the K matrices $\boldsymbol{M}^{(k)}$ by $\boldsymbol{H}(-\phi_m^{n_s}) \boldsymbol{M}^{(k)} \boldsymbol{H}(\phi_m^{n_s})$;
12: Replace $\boldsymbol{A}$ by $\boldsymbol{A}\boldsymbol{G}(\theta_m^{n_s})\boldsymbol{H}(\phi_m^{n_s})$;
13: $m = m + 1$;
14: **end for**
15: **end for**
16: $n_s = n_s + 1$;
17: **end while**
18: $N_s = n_s$;

2.3. Computational complexity

The computational complexity of an algorithm is given by the number Γ of floating point operations (flop), given in practice by the number of required multiplications. At each sweep, there are $N(N-1)/2$ Givens and hyperbolic matrices to compute and as many updates of matrices $\boldsymbol{A}, \boldsymbol{M}^{(1)}, \cdots, \boldsymbol{M}^{(K)}$. Computation of each hyperbolic matrix is dominated by the product $\boldsymbol{J}\boldsymbol{W}^{(m,n_s)\top}\boldsymbol{W}^{(m,n_s)}$ which requires $3K$ multiplications. Givens matrices are computed in a similar way [42] and thus also need $3K$ multiplications. For each update (line 12 of algorithm 1), matrix $\boldsymbol{A}$ is multiplied by a Givens and a hyperbolic matrix. Both products can be done using a total of $8N$ multiplications. Finaly the update of each matrix $\boldsymbol{M}^{(k)}$ (lines 9 and 11 of algorithm 1) is twice more costly and involves $16N$ multiplications. Therefore the total computational complexity is:

$$\Gamma_{JDTM} = N_s N(N-1)(3K + 4N + 8KN) \tag{25}$$

2.4. Extension to the complex case

Let's now consider that matrices $\boldsymbol{A}$ and $\boldsymbol{M}^{(1)}, \cdots, \boldsymbol{M}^{(K)}$ belong to the complex field. In this case, the JDTM algorithm has to be significantly modified. Indeed, each of the Givens and hyperbolic rotation matrices involved in the polar decomposition of a complex matrix is now defined by two parameters. Similarly to the real case, we only focus on the determination of

hyperbolic matrices $\boldsymbol{H}$ which makes the specificity of the proposed algorithm. Indeed, $\boldsymbol{G}$ can still be estimated by the classic procedure [42].

We resort to the following classical parametrization of complex hyperbolic matrices, for each couple $m = (i, j)_{i<j}$ we have:

$$H(\phi_m, \alpha_m)_{ii} = H(\phi_m, \alpha_m)_{jj} = \cosh(\phi_m); \quad H(\phi_m, \alpha_m)_{ij} = \overline{H}(\phi_m, \alpha_m)_{ji} = \sinh(\phi_m)\mathrm{e}^{i\alpha_m}$$

Thereby we have to estimate for each matrix the couple (ϕ_{ij}, α_{ij}) that minimizes the new JDTM cost function:

$$\zeta_{HC}^{JDTM}(\phi_m^{n_s}, \alpha_m^{n_s}) = \sum_{k=1}^{K} |M_{ij}^{(k,m,n_s)}|^2 + |M_{ji}^{(k,m,n_s)}|^2. \tag{26}$$

Using the previous parametrization, we obtain:

$$M_{ij}^{(k,m,n_s)} = \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \frac{\sinh(2\phi_m^{n_s})}{2} \mathrm{e}^{-i\alpha_m^{n_s}} + N_{ij}^{(k,m,n_s)} \cosh(\phi_m^{n_s})^2 - N_{ji}^{(k,m,n_s)} \sinh(\phi_m^{n_s})^2 \mathrm{e}^{-2i\alpha_m^{n_s}}, \tag{27}$$

$$M_{ji}^{(k,m,n_s)} = \left(N_{jj}^{(k,m,n_s)} - N_{ii}^{(k,m,n_s)}\right) \frac{\sinh(2\phi_m^{n_s})}{2} \mathrm{e}^{i\alpha_m^{n_s}} - N_{ij}^{(k,m,n_s)} \sinh(\phi_m^{n_s})^2 \mathrm{e}^{2i\alpha_m^{n_s}} + N_{ji}^{(k,m,n_s)} \cosh(\phi_m^{n_s})^2. \tag{28}$$

It can be easily shown that minimizing ζ_{HC}^{JDTM} is equivalent to minimizing $\tilde{\zeta}_{HC}^{JDTM}$:

$$\tilde{\zeta}_{HC}^{JDTM}(\phi_m^{n_s}, \alpha_m^{n_s}) = \sum_{k=1}^{K} |\tilde{M}_{ij}^{(k,m,n_s)} + M_{ji}^{(k,m,n_s)}|^2 + |\tilde{M}_{ij}^{(k,m,n_s)} - M_{ji}^{(k,m,n_s)}|^2, \tag{29}$$

where:

$$\tilde{M}_{ij}^{(k,m,n_s)} = \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \frac{\sinh(2\phi_m^{n_s})}{2} \mathrm{e}^{i\alpha_m^{n_s}} + N_{ij}^{(k,m,n_s)} \cosh(\phi_m^{n_s})^2 \mathrm{e}^{2i\alpha_m^{n_s}} - N_{ji}^{(k,m,n_s)} \sinh(\phi_m^{n_s})^2. \tag{30}$$

After some straightforward computations, (28), (29) and (30) yield:

$$\begin{aligned}
\tilde{\zeta}_{HC}^{JDTM}(\phi_m^{n_s}, \alpha_m^{n_s}) = & \sum_{k=1}^{K} \left(\left|N_{ij}^{(k,m,n_s)}\right|^2 + \left|N_{ji}^{(k,m,n_s)}\right|^2\right) \cosh(2\phi_m^{n_s})^2 \\
& + \left(\left|N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right|^2 - \left(N_{ij}^{(k,m,n_s)} \overline{N}_{ji}^{(k,m,n_s)} \mathrm{e}^{2i\alpha_m^{n_s}} + N_{ij}^{(k,m,n_s)} \overline{N}_{ij}^{(k,m,n_s)} \mathrm{e}^{-2i\alpha_m^{n_s}}\right)\right) \sinh(2\phi_m^{n_s})^2 \\
& + \tfrac{1}{2}\left(\overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)} N_{ij}^{(k,m,n_s)} - \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ji}^{(k,m,n_s)}\right) \mathrm{e}^{i\alpha_m^{n_s}} \sinh(4\phi_m^{n_s}) \\
& + \tfrac{1}{2}\left(\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ij}^{(k,m,n_s)} - N_{ji}^{(k,m,n_s)} \overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)}\right) \mathrm{e}^{-i\alpha_m^{n_s}} \sinh(4\phi_m^{n_s})
\end{aligned} \tag{31}$$

which can be rewritten as a function of $4\phi_m^{n_s}$ and $\alpha_m^{n_s}$:

$$\begin{aligned}
\tilde{\zeta}_{HC}^{JDTM}(4\phi_m^{n_s}, \alpha_m^{n_s}) = & \tfrac{1}{2} \sum_{k=1}^{K} \left(\left|N_{ij}^{(k,m,n_s)}\right|^2 + \left|N_{ji}^{(k,m,n_s)}\right|^2\right) \left(\cosh(4\phi_m^{n_s}) + 1\right) \\
& + \left(\left|N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right|^2 - \left(N_{ij}^{(k,m,n_s)} \overline{N}_{ji}^{(k,m,n_s)} \mathrm{e}^{2i\alpha_m^{n_s}} + N_{ij}^{(k,m,n_s)} \overline{N}_{ij}^{(k,m,n_s)} \mathrm{e}^{-2i\alpha_m^{n_s}}\right)\right) \left(\cosh(4\phi_m^{n_s}) - 1\right) \\
& + \left(\overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)} N_{ij}^{(k,m,n_s)} - \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ji}^{(k,m,n_s)}\right) \mathrm{e}^{i\alpha_m^{n_s}} \sinh(4\phi_m^{n_s}) \\
& + \left(\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ij}^{(k,m,n_s)} - N_{ji}^{(k,m,n_s)} \overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)}\right) \mathrm{e}^{-i\alpha_m^{n_s}} \sinh(4\phi_m^{n_s}).
\end{aligned} \tag{32}$$

Differentiating (32) with respect to $4\phi_m^{n_s}$ and $\alpha_m^{n_s}$ alternatively, then defining $t_m^{n_s} = \tanh(2\phi_m^{n_s})$ and $z_m^{n_s} = \mathrm{e}^{i\alpha_m^{n_s}}$, it can be shown after few more trivial computations that the solution couple which minimizes $\tilde{\zeta}_{HC}^{JDTM}$ is also a solution of the following polynomial system:

$$\begin{cases} P_0(z_m^{n_s}) + (2P_1(z_m^{n_s})t_m^{n_s} + P_0(z_m^{n_s})t_m^{n_s})t_m^{n_s} = 0 & (33) \\ (Q_1(z_m^{n_s})t_m^{n_s} - Q_0(z_m^{n_s}))t_m^{n_s} = 0 & (34) \end{cases}$$

with:

$$\begin{aligned} P_0(z) = & \sum_{k=1}^{K} \left(\overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)} N_{ij}^{(k,m,n_s)} - \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ji}^{(k,m,n_s)} \right) z^3 \\ & + \left(\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ij}^{(k,m,n_s)} - N_{ji}^{(k,m,n_s)} \overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)} \right) z \\ P_1(z) = & \sum_{k=1}^{K} -\overline{N}_{ji}^{(k,m,n_s)} N_{ij}^{(k,m,n_s)} z^4 + \left(\left|N_{ij}^{(k,m,n_s)}\right|^2 + \left|N_{ji}^{(k,m,n_s)}\right|^2 + \left|N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right|^2 \right) z^2 - \overline{N}_{ij}^{(k,m,n_s)} N_{ji}^{(k,m,n_s)} \\ Q_0(z) = & \sum_{k=1}^{K} \left(\overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)} N_{ij}^{(k,m,n_s)} - \left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ji}^{(k,m,n_s)} \right) z^3 \\ & - \left(\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right) \overline{N}_{ij}^{(k,m,n_s)} - N_{ji}^{(k,m,n_s)} \overline{\left(N_{ii}^{(k,m,n_s)} - N_{jj}^{(k,m,n_s)}\right)} \right) z \\ Q_1(z) = & \sum_{k=1}^{K} 2 \left(\overline{N}_{ji}^{(k,m,n_s)} N_{ij}^{(k,m,n_s)} z^4 - \overline{N}_{ij}^{(k,m,n_s)} N_{ji}^{(k,m,n_s)} \right) \end{aligned} \quad (35)$$

Solution sets are then easily given by:

$$P_0(z_m^{n_s}) = 0 \text{ and } t_m^{n_s} = 0; \quad (36)$$

or:

$$P_0(z_m^{n_s})(Q_1(z_m^{n_s}))^2 + 2P_1(z_m^{n_s})Q_0(z_m^{n_s})Q_1(z_m^{n_s}) + P_0(z_m^{n_s})(Q_0(z_m^{n_s}))^2 = 0 \text{ and } t_m^{n_s} = \frac{Q_0(z_m^{n_s})}{Q_1(z_m^{n_s})}. \quad (37)$$

3. Toward a new direct CPD algorithm: the DIAG method

3.1. The Canonical Polyadic Decomposition

CPD states that any Q-order tensor (or Q-way array) $\boldsymbol{\mathcal{T}}$ of size $I_1 \times \cdots \times I_Q$ can be exactly decomposed into a sum of Q-order rank-1 tensors. A Q-order rank-1 tensor can be defined as the outer product between Q vectors $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(Q)}$. The rank R of $\boldsymbol{\mathcal{T}}$ is then the minimal number of rank-1 tensors needed to achieve the following decomposition:

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \boldsymbol{x}_r^{(1)} \circ \cdots \circ \boldsymbol{x}_r^{(Q)}. \quad (38)$$

Usually one also defines Q "loading" (or factor) matrices $\boldsymbol{X}^{(1)}, \cdots, \boldsymbol{X}^{(Q)}$ of size $I_1 \times R$, $\cdots$, $I_Q \times R$, respectively, so that $\boldsymbol{x}_r^{(q)}$ is the r^{th} column of $\boldsymbol{X}^{(q)}$ and the CPD is commonly rewritten as:

$$\forall q \in [1; Q]_{\mathbb{N}},\ \forall i_q \in [1; I_q]_{\mathbb{N}},\ \mathcal{T}_{i_1 \cdots i_Q} = \sum_{r=1}^{R} X^{(1)}_{i_1 r} X^{(2)}_{i_2 r} \cdots X^{(Q)}_{i_Q r}. \tag{39}$$

Our main problem is thus to find for a given tensor $\boldsymbol{\mathcal{T}}$ of given rank R and order Q, the Q factor matrices that solves (39).

3.2. Unfolding matrix

It is well known that the CPD can be rewritten in a matrix form. Indeed, the tensor dimensions can be merged in order to store all tensor entries in a single "unfolding" matrix. Obviously, there are many way to merge the tensor dimensions and thus many possible unfolding matrices. As it will be seen, the choice of the unfolding matrix has an impact on the algorithm limitations and performance. Therefore, in order to cover all the possibilities, we introduce a P parameter in order that the P first dimensions are merged into the matrix rows whereas the remaining $Q - P$ dimensions are merged into the matrix columns. The corresponding unfolding matrix is denoted by $\boldsymbol{T}(P)$. Note that all the other unfolding matrices can be merely obtained by permuting the tensor dimensions and changing the P value. $\boldsymbol{T}(P)$ entries are linked to $\boldsymbol{\mathcal{T}}$ entries by the following transfer formulas:

$$\forall (m, n) \in [1; \pi_1^P]_{\mathbb{N}} \times [1; \pi_{P+1}^Q]_{\mathbb{N}},\ \boldsymbol{T}(P)_{m,n} = \boldsymbol{\mathcal{T}}_{i_1, \cdots, i_Q} \tag{40}$$

where, $\pi_a^a = I_a$, $\pi_a^b = I_a I_{a+1} \cdots I_b$ and:

$$\forall m \in [1; \pi_1^P]_{\mathbb{N}}, \quad m = i_1 + \sum_{q=2}^{P} (i_q - 1)\pi_1^{q-1}, \tag{41}$$

$$\forall n \in [1; \pi_{P+1}^Q]_{\mathbb{N}}, \quad n = i_{P+1} + \sum_{q=P+2}^{Q} (i_q - 1)\pi_{P+1}^{q-1}. \tag{42}$$

Then after some computations the CPD equation (39) can be rewritten as:

$$\boldsymbol{T}(P) = \left(\boldsymbol{X}^{(P)} \odot \cdots \odot \boldsymbol{X}^{(1)}\right)\left(\boldsymbol{X}^{(Q)} \odot \cdots \odot \boldsymbol{X}^{(P+1)}\right)^{\mathrm{T}}. \tag{43}$$

It is worth mentioning that a majority of CPD algorithms such as ALS or CFS resorts to the $P = 1$ case.

3.3. The DIAG algorithm

The algorithm presented here is available both in the real and complex field. We start from equation (43) and we define for a given couple of integers a and b, $a < b$, the matrix $\boldsymbol{Y}_X^{(b,a)}$ by:

$$\boldsymbol{Y}_X^{(b,a)} = \boldsymbol{X}^{(b)} \odot \cdots \odot \boldsymbol{X}^{(a)}. \tag{44}$$

Now, let $\boldsymbol{USV}^{\mathrm{H}}$ be the singular value decomposition of $\boldsymbol{T}(P)$ truncated at the order R, assuming that $R \leq \min(\pi_1^P, \pi_{P+1}^Q)$ (hypothesis $\mathcal{H}_1$). Thus there exists an invertible square matrix $\boldsymbol{M}$ of size $R \times R$ such that:

$$\boldsymbol{Y}_X^{(P,1)} = \boldsymbol{UM}, \tag{45}$$

$$\boldsymbol{Y}_X^{(Q,P+1)_{\mathrm{T}}} = \boldsymbol{M}^{-1}\boldsymbol{SV}^{\mathrm{H}}. \tag{46}$$

Recalling that $\boldsymbol{Y}_X^{(Q,P+1)} = \boldsymbol{X}^{(Q)} \odot \boldsymbol{Y}_X^{(Q-1,P+1)}$ and using the definition of the Kathri-Rao product, $\boldsymbol{Y}_X^{(Q,P+1)_\intercal}$ can be seen as a row block matrix:

$$\boldsymbol{Y}_X^{(Q,P+1)_\intercal} = \left[\boldsymbol{\phi}^{(1)}\boldsymbol{Y}_X^{(Q-1,P+1)_\intercal}, \cdots, \boldsymbol{\phi}^{(I_Q)}\boldsymbol{Y}_X^{(Q-1,P+1)_\intercal}\right], \tag{47}$$

where $\boldsymbol{\phi}^{(1)}, \cdots, \boldsymbol{\phi}^{(I_Q)}$ are the I_Q diagonal matrices built from the I_Q rows of the matrix $\boldsymbol{X}^Q$. As a consequence, equations (46) and (47) yield:

$$\boldsymbol{S}\boldsymbol{V}^\mathsf{H} = \left[\boldsymbol{\Gamma}^{(1)_\intercal}, \cdots \boldsymbol{\Gamma}^{(I_Q)_\intercal}, \right], \tag{48}$$

where :

$$\forall i \in [1; I_Q]_\mathbb{N},\ \boldsymbol{\Gamma}^{(i)} = \boldsymbol{Y}_X^{(Q-1,P+1)}\boldsymbol{\phi}^{(i)}\boldsymbol{M}^\intercal. \tag{49}$$

All matrices $\boldsymbol{\Gamma}^{(i)}$ and $\boldsymbol{Y}_X^{(Q-1,P+1)}$ are of size $\pi_{P+1}^{Q-1} \times R$. We assume that P is chosen so that $P < Q-1$ and $R \leq \pi_{P+1}^{Q-1}$ (hypothesis $\mathcal{H}_2$) and that they all admit a Moore-Penrose matrix inverse. Then we define:

$$\forall i_1, i_2 \in [1; I_Q]_\mathbb{N}^2,\ i_2 > i_1\ \boldsymbol{\Theta}^{(i_1,i_2)} = \boldsymbol{\Gamma}^{(i_1)\sharp}\boldsymbol{\Gamma}^{(i_2)}. \tag{50}$$

Now replacing $\boldsymbol{\Gamma}^{(i)}$ by its definition yields:

$$\begin{aligned} \boldsymbol{\Theta}^{(i_1,i_2)} &= \boldsymbol{M}^{-\intercal}\boldsymbol{\phi}^{(i_1)-1}\boldsymbol{Y}_X^{(Q-1,P+1)\sharp}\boldsymbol{Y}_X^{(Q-1,P+1)}\boldsymbol{\phi}^{(i_2)}\boldsymbol{M}^\intercal, & (51)\\ &= \boldsymbol{M}^{-\intercal}\boldsymbol{\Lambda}^{(i_1,i_2)}\boldsymbol{M}^\intercal, & (52) \end{aligned}$$

where $\boldsymbol{\Lambda}^{(i_1,i_2)} = \boldsymbol{\phi}^{(i_1)-1}\boldsymbol{\phi}^{(i_2)}$. Thus, $\boldsymbol{M}^{-\intercal}$ performs the JEVD of the known set of matrices $\boldsymbol{\Theta}^{(i_1,i_2)}$. Therefore $\boldsymbol{M}^{-\intercal}$ can be estimated by the JDTM algorithm. Then one can immediately deduce $\boldsymbol{Y}_X^{(P,1)}$ and $\boldsymbol{Y}_X^{(Q,P+1)}$ from (45) and (46). At this stage there are several ways to estimate the factor matrices from $\boldsymbol{Y}_X^{(P,1)}$ and $\boldsymbol{Y}_X^{(Q,P+1)}$. One simple approach is to estimate each column of the first P factor matrices from the corresponding column of $\boldsymbol{Y}_X^{(P,1)}$ and each column of the $Q-P$ remaining factor matrices from the corresponding column of $\boldsymbol{Y}_X^{(Q,P+1)}$. Indeed, column r of $\boldsymbol{Y}_X^{(P,1)}$ can be reshaped into an order-P, rank-1 tensor $\boldsymbol{\mathcal{Y}}_{Xr}^{(P,1)}$ whose factor vectors are the r-th columns of matrices $\boldsymbol{X}^{(1)}, \cdots, \boldsymbol{X}^{(P)}$. Thereby a simple rank-1 High-Order SVD (HOSVD, [43]) of $\boldsymbol{\mathcal{Y}}_{Xr}^{(P,1)}$ provides a direct estimation of $\boldsymbol{x}_r^{(1)}, \cdots, \boldsymbol{x}_r^{(P)}$. In the same way, the column r of $\boldsymbol{Y}_X^{(Q,P+1)}$ can be reshaped in a $(Q-P)$-order, rank-1 tensor $\boldsymbol{\mathcal{Y}}_{Xr}^{(Q,P+1)}$ whose factor vectors are the r-th columns of matrices $\boldsymbol{X}^{(P+1)} \cdots \boldsymbol{X}^{(Q)}$. Hence, $\boldsymbol{x}_r^{(P+1)} \cdots \boldsymbol{x}_r^{(Q)}$ can be estimated from the rank-1 HOSVD of $\boldsymbol{\mathcal{Y}}_{Xr}^{(Q,P+1)}$. Finally both operations are repeated for all the r values. The DIAG algorithm is summarized by Algorithm 2.

3.4. Computational complexity

Γ_{DIAG} is clearly dominated by the three following computations. First, the truncated SVD of the unfolding matrix of size (π_1^P, π_{P+1}^Q) requires $2\pi_{P+1}^Q(\pi_1^P)^2 + 5R^2(\pi_1^P + \pi_{P+1}^Q) - 2(R^3 + (\pi_1^P)^3)/3$ multiplications, assuming that $\pi_{P+1}^Q > \pi_1^P$. Then, the computation of the $\boldsymbol{\Theta}$ matrices needs approximately $(RI_Q)^2\pi_{P+1}^{Q-1}$ additional multiplications. Finally the cost of the JEVD procedure is approximated by $8N_s(I_Q)^2R^3$. Additional computations can be neglected and thus we have:

$$\Gamma_{DIAG} \approx 2\pi_{P+1}^Q(\pi_1^P)^2 + 5R^2(\pi_1^P + \pi_{P+1}^Q) - 2(R^3 + (\pi_1^P)^3)/3 + (RI_Q)^2\pi_{P+1}^{Q-1} + 8N_s(I_Q)^2R^3. \tag{53}$$

Algorithm 2: Summary of the DIAG algorithm

Choose a value of P and a permutation of the dimensions of $\mathcal{T}$ as described in section 3.6;
Matricize the (possibly permuted) tensor $\mathcal{T}$ into matrix $\boldsymbol{T}(P)$ according to (40), (41) and (42);
Compute the SVD $\boldsymbol{USV}^{\text{H}}$ of $\boldsymbol{T}(P)$, truncated at rank R;
Split $\boldsymbol{SV}^{\text{H}}$ into I_Q blocks of size $R \times \pi_{P+1}^{Q-1}$ in order to form the I_Q matrices $\Gamma^{(i)}$ given by (49);
for $i_1 = 1$ to $I_Q - 1$ **do**
  for $i_2 = i_1 + 1$ to I_Q **do**
    Compute $\boldsymbol{\Theta}^{(i_1,i_2)} = \boldsymbol{\Gamma}^{(i_1)\sharp}\boldsymbol{\Gamma}^{(i_2)}$;
  end for
end for
Compute matrix $\boldsymbol{M}^{-\text{T}}$ by JEVD of the set of $\boldsymbol{\Theta}^{(i_1,i_2)}$ matrices;
Deduce matrices $\boldsymbol{Y}_X^{(P,1)} = \boldsymbol{UM}$ and $\boldsymbol{Y}_X^{(Q,P+1)} = \boldsymbol{M}^{-1}\boldsymbol{SV}^{\text{H}}$;
for $r = 1$ to R **do**
  Build $\boldsymbol{\mathcal{Y}}_{Xr}^{(P,1)}$ and $\boldsymbol{\mathcal{Y}}_{Xr}^{(Q,P+1)}$ by reshaping the $r-$th columns of $\boldsymbol{Y}_X^{(P,1)}$ and $\boldsymbol{Y}_X^{(Q,P+1)}$;
  Deduce $\boldsymbol{x}_r^{(1)}, \cdots, \boldsymbol{x}_r^{(P)}$ from the rank 1 HOSVD of $\boldsymbol{\mathcal{Y}}_{Xr}^{(P,1)}$;
  Deduce $\boldsymbol{x}_r^{(P+1)}, \cdots, \boldsymbol{x}_r^{(Q)}$ the rank 1 HOSVD of $\boldsymbol{\mathcal{Y}}_{Xr}^{(Q,P+1)}$;
end for

Γ_{DIAG} should be compared to the numerical complexity of the ALS algorithm which is approximately given by:

$$\Gamma_{ALS} \approx N_{ALS}\left(3R\pi_1^Q + 7R^2\sum_{q=1}^{Q}\prod_{\substack{k=1\\k\neq q}}^{Q} I_k\right), \tag{54}$$

However the numerical complexity of the DIAG algorithm is strongly related to the choice of the unfolding matrix and both complexities depend on a large number of parameters. Furthermore N_{ALS} can fluctuate wildly. Therefore at this point it would be very hazardous to draw general conclusions from the previous formulas even in simple cases. Nevertheless we made some extensive flop comparisons between both algorithms by varying Q, R, P and the tensor dimensions. Results are reported in section 4.2.4. It will be shown that in all the considered situations $\Gamma_{DIAG} \leq \Gamma_{ALS}$ and $N_s \ll N_{ALS}$.

The numerical complexity of the CFS algorithm is very complicated to establish since this algorithm computes several estimations of each factor matrix. However we can easily explain what makes DIAG a cheaper approach. CFS is a three step algorithm. The first step is algebraic and performs the HOSVD of the tensor. In terms of numerical complexity this operation is usually close to the SVD of the unfolding matrix performed in the DIAG algorithm. The second step is the resolution of $Q(Q-1)^2$ JEVDs whereas DIAG requires only one JEVD. Finally, we have to choose the best estimates of the factor matrices among a large number of combinations which is also very time consuming.

3.5. Necessary conditions to the identifiability of DIAG, ALS and CFS

The CPD algorithms are not always applicable due to their intrinsic restricted conditions. We propose to compare here necessary conditions that ensure identifiability of the ALS, CFS and

DIAG methods. Let Q, R and $I(i)$ be the tensor order, the CPD rank and the i-th dimension of the tensor, respectively. A tensor of order Q and rank R can be canonically decomposed by ALS only if:

$$(C_{ALS}): \ \forall\, q \in [1;Q]_{\mathbb{N}},\ \prod_{\substack{i=1\\ i\neq q}}^{Q} I(i) \geq R. \tag{55}$$

DIAG conditions are given by hypotheses $\mathcal{H}_1$ and $\mathcal{H}_2$. $\mathcal{H}_1$ and $\mathcal{H}_2$ were expounded for a given order of the tensor dimensions (default order). Actually, By taking into account that the dimensions can be permuted we obtain the following more general condition:

$$(C_{DIAG}): \quad \exists\, P \in [2;Q-1]_{\mathbb{N}}, \exists\ f_I \text{ a permutation of the } Q \text{ first natural numbers and } \exists\, q_s > P \text{ such that:} \\ \prod_{i=1}^{P} I(f_I(i)) \geq R \text{ and } \prod_{\substack{i=P+1\\ i\neq q_s}}^{Q} I(f_I(i)) \geq R. \tag{56}$$

Finally, the condition C_{CFS} for the closed-form solution is given in [28]:

$$(C_{CFS}): \exists\, (q_1,q_2) \in [1;Q]^2_{\mathbb{N}},\ q_1 \neq q_2 \text{ such that } I(q_1) \geq R \text{ and } I(q_2) \geq R. \tag{57}$$

Proposition 1. *C_{DIAG} is more restrictive than C_{ALS} but less restrictive than C_{CFS} :*
$C_{CFS} \Rightarrow C_{DIAG} \Rightarrow C_{ALS}$

A proof is given in appendix. In practice the DIAG condition implies $P \leq Q-2$ and can be reformulated quite easily for low order tensors ($3 \leq Q \leq 5$):

Third order tensors, $Q = 3$**.** Here we have necessarily $P = 1$ hence C_{DIAG} becomes simply: at least two of the tensor dimensions are greater or equal to the CPD rank R. Thereby at order 3 (and only at order 3) C_{DIAG} and C_{CFS} are equivalent.

Fourth order tensors, $Q = 4$**.** Here we can choose either $P = 1$ or $P = 2$ but the condition remains the same in both cases and is simply: at least one tensor dimension is greater than R and at least one product of two of the remaining dimensions is also greater than R.

Fifth order tensors, $Q = 5$**.** Here $1 \leq P \leq 3$:

- if we choose $P = 1$ or $P = 3$ then C_{DIAG} becomes: at least one tensor dimension is greater than R and at least one product of three of the remaining dimensions is also greater than R.
- if we choose $P = 2$ then C_{DIAG} becomes: at least one product between two tensor dimensions and another product between two of the remaining dimensions are greater than R.

3.6. Choice of the unfolding matrix

An obvious criterion is the residual error between $\mathcal{T}$ and the reconstructed tensor built from the estimated factor matrices. However it would be very time consuming to test several possibilities. As a consequence the choice of the more appropriate unfolding matrix should be related to hypothesis $\mathcal{H}_1$ and $\mathcal{H}_2$. Indeed, one has to choose a permutation of the tensor dimensions and a P value that ensure both hypotheses. Otherwise, the DIAG algorithm is not suitable as it is explained in the previous section. Recall notably that the DIAG algorithm implies $P \leq Q-2$.

Indeed, at order 3 we have necessarily $P = 1$. At order 4 we have two possible values (1 and 2) and so on. Therefore if one wants to maximize the value of the highest possible rank then one should maximize $\min(\pi_1^{q-1}, \pi_{P+1}^{q-1})$, hence choose $\boldsymbol{T}(p)$ as squared as possible. In practice we observed that this recommendation is always a good option even if all tensor dimensions are greater than the rank. Apart from that one should note that the number of matrices to be jointly diagonalized is directly related to the squared dimension of the last mode and thus the numerical complexity of the JEVD step. Therefore in the case of a tensor with one very large dimension we do not recommend to put it at the end (if possible). More generally, we recommend to take into consideration the overall complexity of the DIAG algorithm given by equation (53) and to consider that with the JDTM algorithm the number of sweeps (N_s) exceeds very rarely 10. In section 4.2.4 we give several significant numerical examples of DIAG complexity for various tensor dimensions and unfolding matrices.

4. Numerical simulations

The proposed algorithms are first validated on synthesized data sets. We first focus the JEVD sub-problem for which we compare JDTM performances to these of other JEVD algorithms. Then we compare the DIAG approach with CFS, an other direct algorithm and ALS-ELS which is a reference iterative method, with respect to several scenarios. The last subsection is dedicated to a particular tensor family for which iterative algorithms consistently fail to find the CPD.

4.1. Performance comparison of the JDTM algorithm

The performance of the JDTM algorithm is studied and compared to that of the JET, sh-rt and JUST methods by varying the number K of matrices to be jointly diagonalized, the Signal-to-Noise Ratio (SNR) and the matrix dimensions N. The matrix set to be jointly diagonalized is built according to the following model:

$$\forall k \in [1; K]_{\mathbb{N}},\ \boldsymbol{M}^{(k)} = \frac{\widetilde{\boldsymbol{M}}^{(k)}}{||\widetilde{\boldsymbol{M}}^{(k)}||_F} + \sigma \frac{\boldsymbol{E}^{(k)}}{||\boldsymbol{E}^{(k)}||_F} \text{ with } \tilde{\boldsymbol{M}}^{(k)} = \boldsymbol{A}\boldsymbol{D}^{(k)}\boldsymbol{A}^{-1}. \tag{58}$$

Entries of $\boldsymbol{A}$, $\boldsymbol{D}^{(k)}$ and $\boldsymbol{E}^{(k)}$ are drawn randomly according to a standard normal distribution. The scalar parameter σ allows us to regulate the power of the Gaussian additive noise $\boldsymbol{E}^{(k)}$. The SNR is then equal to $-20\log_{10}(\sigma)$. Hence, σ is chosen in order to obtained the desired value of SNR.

At the end of each sweep, the squared off-diagonal components of the K matrices $\boldsymbol{M}^{(k,M,n_s)}$ are summed and the obtained value is compared to the value computed at the previous sweep. Algorithms are stopped when the relative deviation between two successive values is smaller than 10^{-3}.

After having removed the scaling and permutation indeterminacies we define r_A as the relative root squared error between the true eigenvector matrix and its estimate $\widehat{\boldsymbol{A}}$:

$$r_A = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(A_{i,j} - \widehat{A}_{i,j}\right)^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(A_{i,j}\right)^2}}. \tag{59}$$

Note that in most practical applications and notably in blind source separation, one is only interested by the estimation of the eigenvector matrix. Hence r_A appears as a relevant JEVD criterion.

Finally the number of sweeps, N_s, required by each algorithm is stored in order to compute the values of the total numerical complexities Γ. Therefore, algorithm results are judged according to three criteria, namely N_s, Γ and r_A.

Each simulation is repeated 100 times with a new draw of the matrices $\boldsymbol{A}$, $\boldsymbol{D}^{(k)}$ and $\boldsymbol{E}^{(k)}$ at each time. We present here median values of r_A and mean values of Γ and N_s obtained from each algorithm.

Figures 1, 2 and 3 show simulation results for 3 SNR values (60 dB, 40 dB and 20 dB respectively). The number of matrices to be jointly diagonalized was fixed to $K = 64$ whereas we varied the matrix size N from 2 to 32. We first note that the estimation precision of the algorithms logically increases with the ratio K/N and the SNR. Second, according to r_A criterion JUST algorithm is consistently outperformed by other algorithms whatever the considered situation. At 60 dB, figure 1(a) points out that the JDTM and JET algorithm outclass the sh-rt approach concerning the estimation of eigenvectors matrix. According to this r_A criterion JET performs slightly better than JDTM for matrix size lower or equal to 16 whereas for the largest size JDTM clearly provides the best performances. The comparison of the average computational costs displayed in figure 1(b) shows very closed results between all the algorithms. However JDTM appears more clearly as the less costly algorithm for largest matrix sizes. This is explained by a lower and remarkably stable number of sweeps (figure 1(c)). Previous conclusions hold at 40 dB. However it is interesting to note that concerning the estimation of the eigenvectors matrix JDTM is now significantly more accurate than JET for $N = 16$ and $N = 32$. Finally, the 20 dB case highlights the efficiency of the JDTM algorithm which clearly improves JET and sh-rt results, for matrix sizes larger than 8. However JET is now the faster algorithm. In conclusion JDTM appears as a very versatile algorithm which provide very accurate results in all the considered situation (in comparison to its competitors) for a lower number of sweeps. This number is remarkably stable, being comprised between 3 and 10 in all the considered scenarios. Moreover JDTM consistently provides the best estimate of the eigenvector matrix for the largest matrix size and this gap increases with the SNR. To sum up, JDTM offers quite similar performances than its best competitors (sh-rt or JET) in the easiest cases (regarding SNR and K/N ratio) whereas it clearly becomes the better choice as the difficulty increases.

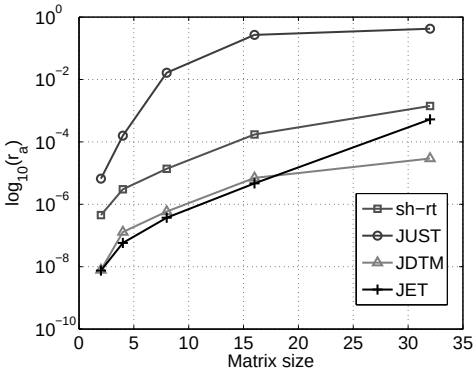
As part of this study, we also evaluate JDTM ability to deal with an ill-conditioned eigenvector matrix. For this purpose, we now compute the eigenvector matrix $\boldsymbol{A}$ with pairwise correlated columns as follows: odd columns, $\boldsymbol{a}_{2r-1}$, are still randomly drawn as previously but even columns, $\boldsymbol{a}_{2r}$, are built in the following way :

$$\forall r \in [1; N/2]_{\mathbb{N}}, \qquad \boldsymbol{a}_{2r} = \nu \boldsymbol{a}_{2r-1} + (1-\nu)\boldsymbol{n}_r, \tag{60}$$

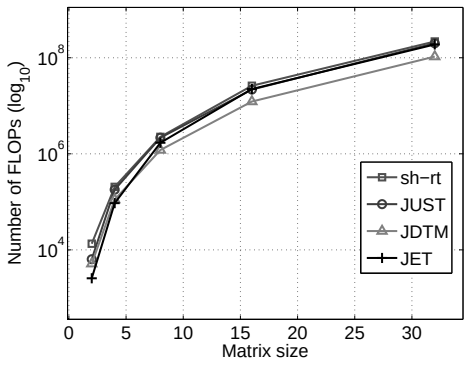
where $\boldsymbol{n}_r$ is a vector of $\mathbb{R}^N$ whose components are randomly drawn according to a standard normal distribution. Thereby ν defines a collinearity factor which will vary from 0.1 to 0.9 so that matrices $\boldsymbol{A}$ can be very ill-conditioned. Figure 4 shows simulation results for a set of 10 matrices of size 10 ($K = N = 10$) at 80 dB. It can be seen that sh-rt, JDTM and JET perform well for $\nu < 0.9$. JDTM and JET provide the best results in terms of estimation precision but JDTM requires a minimal number of sweeps and computational cost.

4.2. Performance comparison of the DIAG algorithm

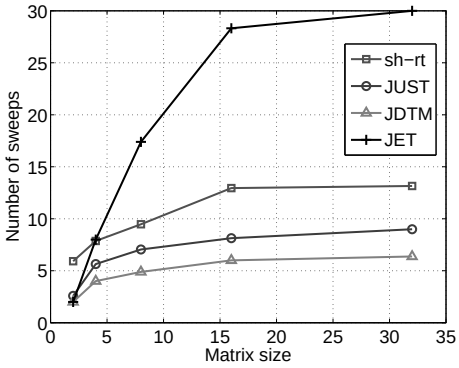
We now study performances of the DIAG algorithm for the decomposition of noisy tensors. Indeed, in most practical applications involving tensor analysis, a noisy tensor of rank R is mod-



(a) r_A criterion (median)

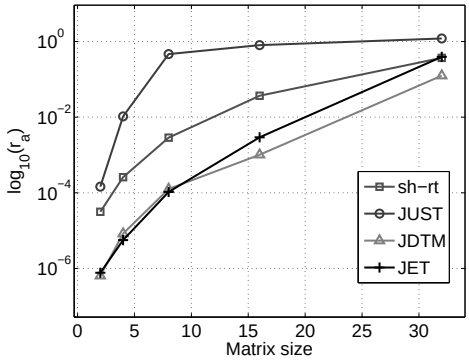


(b) Γ criterion (mean)

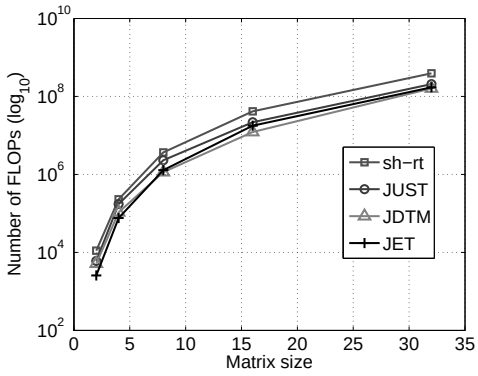


(c) Number of Sweeps (mean)

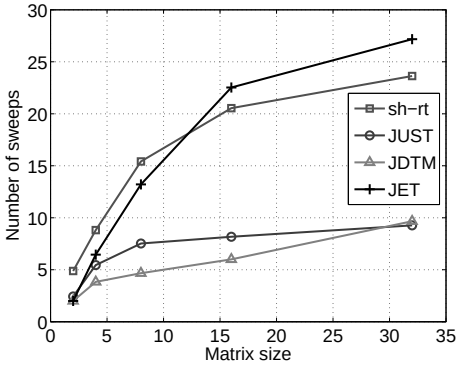
Figure 1: Evolution of the three comparison criteria as a function of the matrix size for a set of 64 matrices with an SNR value of 60 dB.



(a) r_A criterion (median)

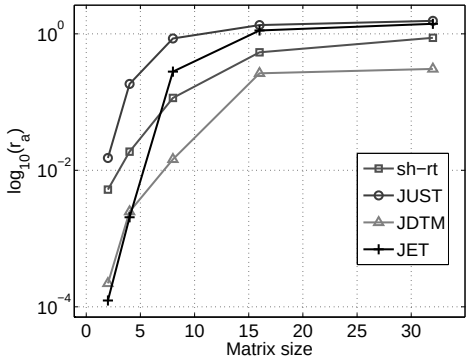


(b) Γ criterion (mean)

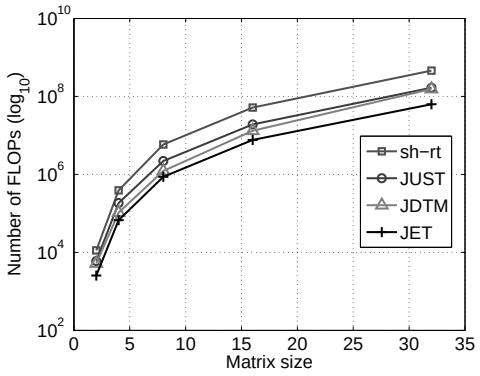


(c) Number of Sweeps (mean)

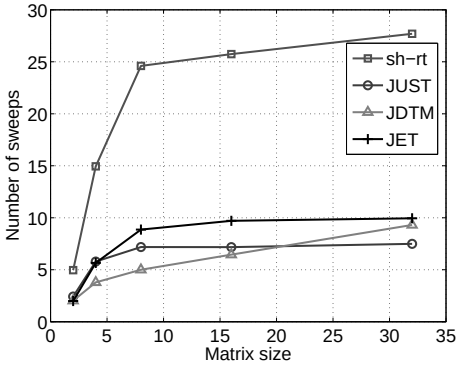
Figure 2: Evolution of the three comparison criteria as a function of the matrix size for a set of 64 matrices with an SNR value of 40 dB.



(a) r_A criterion (median)

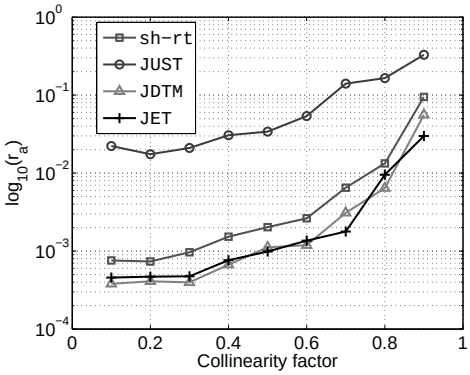


(b) Γ criterion (mean)

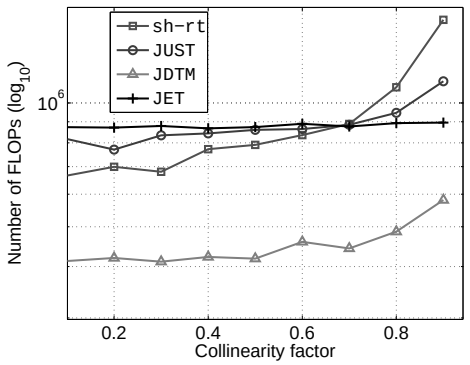


(c) Number of Sweeps (mean)

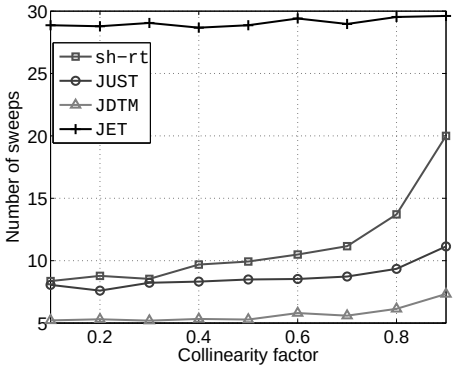
Figure 3: Evolution of the three comparison criteria as a function of the matrix size for a set of 64 matrices with an SNR value of 20 dB.



(a) r_A criterion (median)



(b) Γ criterion (mean)



(c) Number of Sweeps (mean)

Figure 4: Evolution of the three comparison criteria as a function of the correlated factor between columns of matrix $\boldsymbol{A}$ for a set of 10 matrices of size 10 and an SNR value of 80 dB.

elized by "truncated" CPD of rank $R_m < R$ which is usually more relevant than the exact CPD:

$$\forall q \in [1;Q]_{\mathbb{N}},\ \forall i_q \in [1;I_q]_{\mathbb{N}},\ \boldsymbol{\mathcal{T}}_{i_1,\cdots,i_Q} = \sum_{r=1}^{R_m} \boldsymbol{X}^{(1)}_{i_1,r}\boldsymbol{X}^{(2)}_{i_2,r}\cdots\boldsymbol{X}^{(Q)}_{i_Q,r} + \boldsymbol{\mathcal{E}}_{i_1,\cdots,i_Q}, \tag{61}$$

where $\boldsymbol{\mathcal{E}}$ is an error term. R_m is the model rank. The DIAG algorithm is compared with an ALS-ELS algorithm and with the CFS algorithm in various situations by means of Monte-Carlo experiments. For each new experiment, a noise free tensor is built from factor matrices of R_m columns whose entries are randomly drawn according to a standard normal distribution. We then add a Gaussian white noise whose the power is regulated according to the desired SNR value. The comparison criterion, r_X, is the Normalized Mean Squares Error (NMSE) computed between actual and estimated factor matrices. Hence for a tensor of order Q we have:

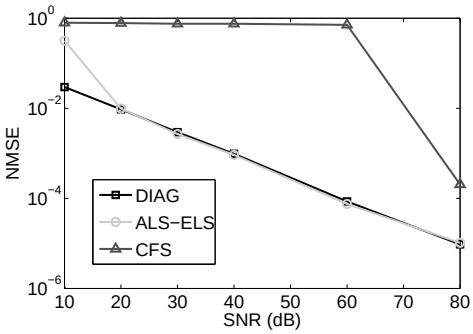
$$r_X = \frac{1}{Q}\sum_{q=1}^{Q} \text{med}\left(\sqrt{\frac{\text{vec}(\boldsymbol{X}^{(q)} - \widehat{\boldsymbol{X}^{(q)}})^{\intercal}\text{vec}(\boldsymbol{X}^{(q)} - \widehat{\boldsymbol{X}^{(q)}})}{\text{vec}(\boldsymbol{X}^{(q)})^{\intercal}\text{vec}(\boldsymbol{X}^{(q)})}}\right), \tag{62}$$

where $\widehat{\boldsymbol{X}^{(q)}}$ denotes the estimation of the factor matrix $\boldsymbol{X}^{(q)}$, the vec$(\cdot)$ operator maps a matrix to a column vector by stacking its columns one below the other and med$(\cdot)$ denotes the median value computed from 100 MC experiments. Permutation and scaling ambiguities in the estimated factor matrices are fixed in the same manner as in [21]. All algorithms were written in-house. The ALS-ELS algorithm can be found in the tensor package web-page [1]. It is stopped as soon as the relative deviation between two consecutive values of the CPD cost function becomes lower than 10^{-6} or the number of ALS iterations reaches 1000. ELS procedure is run every 5 iterations. For the decomposition of order-3 tensors, we use the CFS algorithm described in [27] with the best matching scheme proposed in section 4.2 of [27] whereas higher order tensors were decomposed using the N-order version described in [28], using the sub-optimal matching rules proposed by the authors. Implemented versions of DIAG and CFS resort to the JDTM algorithm to solve the JEVD problem and are stopped as soon as the relative deviation between two consecutive values of the JEVD cost function becomes lower than 10^{-6} or the number of JEVD iterations reaches 30. Unfolding matrix in the DIAG algorithm is generally chosen to be as squared as possible.
Since the number of test parameters is large, it would be impossible to perform here an exhaustive comparison. As a consequence we have limited ourselves to some key situations which illustrate the main features of the proposed approach : *i.* its ability to decompose high order tensors of high rank, *ii.* tensors with almost collinear factors, *iii.* its insensitivity to over-factoring and *iv.* its low computational complexity.

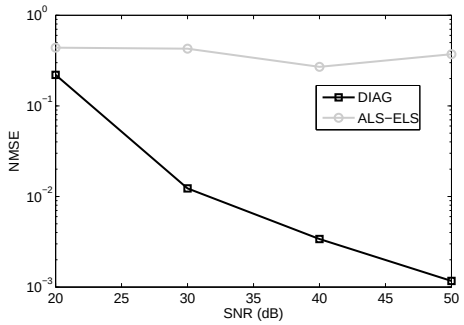
4.2.1. High order tensors

We first consider a set of 6-order tensors of rank 5 whose all the dimensions are equal to 5. DIAG parameter P is set to 3 and we vary the SNR from 10 dB to 80 dB. Results are plotted on figures 5(a). CFS only works for the highest SNR value, probably because this is a difficult situation for which we are very close to its intrinsic limitation. DIAG provides as accurate estimations as ALS-ELS for SNR values greater than 10 dB. ALS-ELS fails at 10 dB while DIAG still works. Notably it clearly outperforms ALS at 10 dB.

[1] `http://www.gipsa-lab.grenoble-inp.fr/~pierre.comon/TensorPackage/tensorPackage.html`



(a) $5 \times 5 \times 5 \times 5 \times 5 \times 5$ tensors of rank 5.



(b) $3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3$ tensors of rank 6.

Figure 5: Median NMSE as a function of the SNR at the output of the ALS and DIAG algorithms applied to high order tensors.

We then consider 8-order tensors of rank 6 whose all the dimensions are equal to 3. For this more difficult case, we vary SNR values from 20 dB to 50 dB. CFS is inapplicable because of its restrictive necessary condition. Indeed tensor rank is larger than the two largest tensor dimensions. P is set to 4. Figures 5(b) shows that in spite of ELS, ALS is usefulness here. Conversely DIAG performs well for the three SNR values above 20 dB.

4.2.2. Influence of the collinearity factor

In the next two following examples we consider the CPD of rank 4 tensors whose columns of the random factor matrices are pairwise correlated in all the modes (swamp). For instance, correlated columns in mode q are built following the scheme of equation (60):

$$\forall r \in [1; R/2]_{\mathbb{N}}\ \boldsymbol{x}_{2r}^{(q)} = \nu \boldsymbol{x}_{2r-1}^{(q)} + (1-\nu)\boldsymbol{n}_r^{(q)}. \tag{63}$$

Note that it has been shown previously in [21] that in this kind of scenarios ALS performances are significantly improved by using ELS. First simulation involves third order tensors of size $4\times4\times4$. For the second simulation we consider fourth order tensors of size $4\times4\times4\times4$. Results are plotted on figures 6(a) and 6(b) respectively. DIAG is the only algorithm which works well in all the considered situations including the most difficult ones (high values of ν) except for $\nu = 0.9$ at order 3. ALS-ELS algorithm fails or is outperformed for largest values of ν ($\nu > 0.5$ at order 3 and $\nu > 0.7$ at order 4). At order 3 CFS results are slightly better than DIAG ones while at order 4 we find an opposite situation when $\nu < 0.7$. When dealing with higher values only DIAG works.

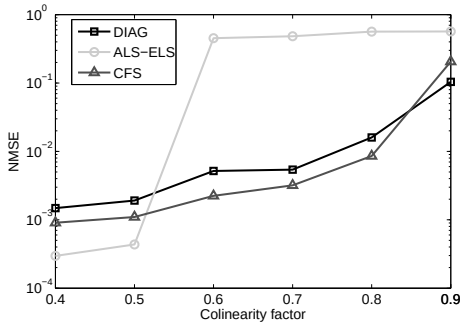
We then perform a third simulation with third order tensors of size $4 \times 4 \times 10 \times 4$. This time all the factors in each mode are mutually correlated:

$$\forall q \in [1; Q]_{\mathbb{N}},\ \forall r \in [2; R]_{\mathbb{N}}\ \boldsymbol{x}_r^{(q)} = \nu \boldsymbol{x}_1^{(q)} + (1-\nu)\boldsymbol{n}_r^{(q)}. \tag{64}$$

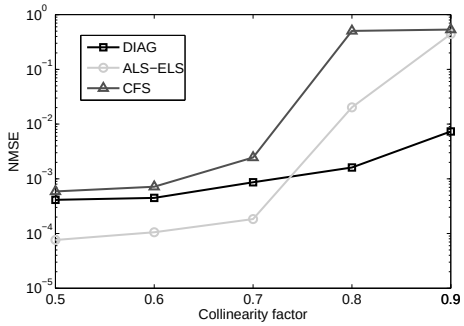
Then we vary tensors rank from 3 to 7 while ν is set to 0.8. This simulation again highlights the main restriction of the CFS algorithm which cannot perform CPD of rank higher than 4. ALS-ELS results are slightly better than DIAG ones for ranks 3 and 4. On the opposite DIAG appears as the best option for higher rank values. Notably it still provide satisfactory results for $R = 7$ contrary to ALS-ELS. Finally we compare the complex version of our algorithm DIAG using the complex JDTM method with the complex version of the ALS algorithm. Complex-valued tensors are built as for the two first examples of this section but using complex-valued factor matrices. We consider here third order tensors of size $5\times5\times5$ and rank 3. Results are displayed in figure 6(d). Results obtained in the complex field are very similar to those obtained in the real field for example 1. Indeed ALS starts to fail for $\nu > 0.4$ whereas DIAG still works at $\nu = 0.9$.

4.2.3. Over-factoring

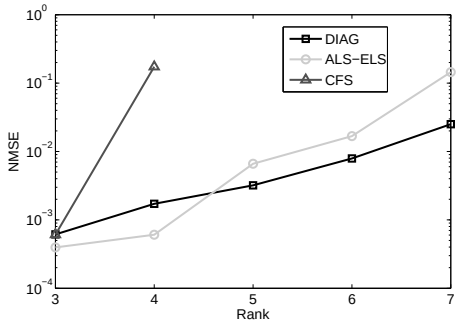
In many practical situations the actual model rank R_m of the data tensor to be decomposed is unknown and it is usually not equal to the tensor rank. Few methods exist for estimating this number. In addition, these sometimes provide ambiguous or contradictory results. This can lead to overestimate the model rank. In other words the corresponding decomposition implies more factors than it is necessary (over-factoring). Suppose that $\widehat{R}_m$ is an overestimation of R_m and Q is the tensor order. A classical problem with ALS is that the $Q(\widehat{R}_m - R_m)$ extra factors not only model the additive noise but also the signal. Hence their estimation affects the estimation of the QR_m actual factors. We study here the impact of over-factoring on DIAG results. For this purpose we successively compute 5 CPD of 3-order noisy tensors of model rank 3 ($R_m = 3, I_1 = I_2 = I_3 = 7$,



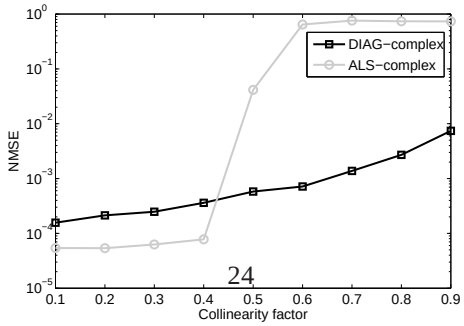
(a) Median NMSE versus collinearity factor for $4 \times 4 \times 4$ tensors of rank 4.



(b) Median NMSE versus collinearity factor for $4 \times 4 \times 4 \times 4$ tensors of rank 4.

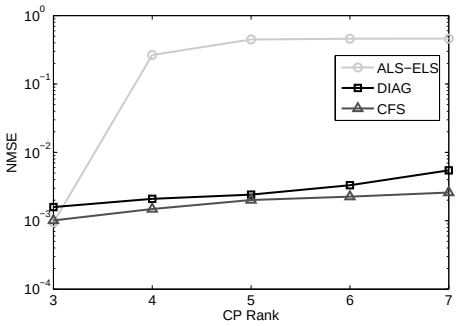


(c) Median NMSE versus tensor rank for $4 \times 4 \times 10 \times 4$ tensors and $\nu = 0.8$.

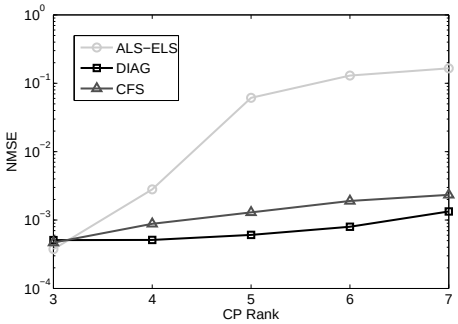


(d) Median NMSE versus tensor rank for $5 \times 5 \times 5$ tensors of rank 3 in the complex field.

Figure 6: Influence of the collinearity factor on the CP decomposition



(a) $7 \times 7 \times 7$ tensors of real rank 3.



(b) $7 \times 7 \times 7 \times 7$ tensors of real rank 3.

Figure 7: Median NMSE as a function of SNR at the output of the ALS-ELS, CFS and DIAG algorithms in the case of over-factoring.

Table 1: Median NMSE, averaged number of iterations and averaged number of flops for small tensors of order 3

Algorithm	$R = 4, I = 10 \times 10 \times 10$			$R = 4, I = 5 \times 100 \times 5$			$R = 4, I = 5 \times 5 \times 100$		
	r_X	N_{it}	Γ	r_X	N_{it}	Γ	r_X	N_{it}	Γ
ALS	2.3×10^{-3}	130	6×10^{6}	2.5×10^{-3}	140	2×10^{7}	2.7×10^{-3}	192	2.8×10^{7}
ALS-ELS	2.3×10^{-3}	47	2×10^{6}	2.5×10^{-3}	46	7.5×10^{6}	2.7×10^{-3}	67	1.1×10^{7}
DIAG	5×10^{-3}	5	3×10^{5}	6.6×10^{-3}	5	1.7×10^{5}	2×10^{-2}	5	2.9×10^{7}
DIAG + ALS-ELS	2.2×10^{-3}	7	6×10^{5}	2.5×10^{-4}	8	1.4×10^{6}	2.7×10^{-3}	9	3×10^{7}

Table 2: Median NMSE, averaged number of iterations and averaged number of flops for large tensors of order 3

Algorithm	$R = 7, I = 50 \times 50 \times 50$			$R = 5, I = 100 \times 100 \times 100$			$R = 4, I = 50 \times 100 \times 50$		
	r_X	N_{it}	Γ	r_X	N_{it}	Γ	r_X	N_{it}	Γ
ALS	5.6×10^{-4}	58	3×10^{8}	2.3×10^{-4}	23	4.6×10^{8}	3.4×10^{-4}	47	2.1×10^{8}
ALS-ELS	5.6×10^{-4}	27	1.8×10^{8}				3.3×10^{-4}	14	8.2×10^{7}
DIAG	2.1×10^{-3}	5	5.5×10^{7}	5.8×10^{-4}	5	2.8×10^{8}	5.7×10^{-4}	5	3.5×10^{7}
DIAG + ALS-ELS	5.4×10^{-4}	4	7.6×10^{7}	2.3×10^{-4}	3	3.3×10^{8}	3.3×10^{-4}	3	4.9×10^{7}

SNR=50 dB) truncated at rank 3 to 7 respectively. After each CPD and for each estimated factor matrix we keep the three columns that best correspond to the actual 3 factors. Thereby at the end of the process we can compute r_X for each CPD. DIAG results are compared with those of ALS-ELS and CFS on figure 7(a). It is worth mentioning that over-factoring has little impact on DIAG and CFS results while ALS-ELS provides incorrect estimations of the actual factors as soon as the model rank is overestimated. This is an important feature of direct approaches. A second simulation is performed in the same way but with 4-order tensors of dimensions $7 \times 7 \times 7 \times 4$. Model rank is still set to 3. Results are plotted on figure 7(b). Again over-factoring strongly affects ALS-ELS estimates. Conversely DIAG and CFS results are consistent even in the case a large number of extra factors is used. We can also note that at order 4 DIAG is less sensitive to over-factoring than CFS.

4.2.4. A trade-off between speed and precision

We have shown some particular situations for which the DIAG algorithm provides the best estimation results. However one of the main advantages of the proposed approach with respect

Table 3: Median NMSE, averaged number of iterations and averaged number of flops for tensors of order 4

Algorithm	$R = 5, I = 5 \times 10 \times 5 \times 10$			$R = 5, I = 5 \times 5 \times 10 \times 10$			$R = 8, I = 5 \times 5 \times 10 \times 5$		
	r_X	N_{it}	Γ	r_X	N_{it}	Γ	r_X	N_{it}	Γ
ALS	1.5×10^{-3}	50	1.5×10^{7}	1.4×10^{-3}	31	9.3×10^{6}	3.5×10^{-3}	11	4.7×10^{7}
ALS-ELS	1.5×10^{-3}	29	9.4×10^{6}	1.4×10^{-3}	23	7.3×10^{6}	3.5×10^{-3}	54	2.4×10^{7}
DIAG	7.1×10^{-3}	5	7.1×10^{5}	3.2×10^{-3}	5	6.5×10^{5}	1.7×10^{-2}	6	6.9×10^{5}
DIAG + ALS-ELS	1.5×10^{-3}	6	2.7×10^{6}	1.4×10^{-3}	6	2.5×10^{6}	3.5×10^{-3}	9	4.6×10^{6}

Table 4: Median NMSE, averaged number of iterations and averaged number of flops for higher order tensor and tensors with correlated factors (all tensors are rank 4)

Algorithm	$I = 7\times7\times7\times7\times7\times7$			$I = 10\times10\times10\times10\times10$			$I = 10\times10\times10$		
	r_X	N_{it}	Γ	r_X	N_{it}	Γ	r_X	N_{it}	Γ
ALS	2.3×10^{-4}	21	2.6×10^{8}	2.5×10^{-4}	28	1.9×10^{8}	4.3×10^{-1}	831	4×10^{7}
ALS-ELS	2.6×10^{-4}	19	2.5×10^{8}				1.2×10^{-3}	444	2×10^{7}
DIAG	2.3×10^{-4}	4	5.4×10^{7}	2.8×10^{-4}	4	2×10^{7}	2.5×10^{-3}	5	3×10^{5}
DIAG + ALS-(ELS)	2.2×10^{-4}	2	8.3×10^{7}	2.5×10^{-4}	2	3.5×10^{7}	6.8×10^{-4}	10	7×10^{5}

to iterative algorithms is its high convergence speed and its lower numerical complexity. Furthermore we still have to evaluate DIAG performances in the case of big tensors. For this purpose, we study here 12 representative examples by varying the tensor dimensions and the CPD rank. Examples are classified into 4 groups of three examples: small tensors of order 3, large tensors of order 3, tensors of order 4 and finally, higher order tensors and tensors with correlated CPD factors. Median NMSE values, averaged numbers of iterations N_{it} and averaged numbers of flops Γ are reported in tables 1, 2, 3 and 4 for each example of the four groups and for an SNR value of 40 dB. DIAG is here compared to ALS and ALS-ELS . DIAG estimates can also be used as initial guests of the ALS-ELS procedure. Hence in these tables, "DIAG + ALS-ELS" refers to the ALS-ELS algorithm initialized with DIAG estimates.

Group of small third order tensors. In the two first examples we show that ALS and ALS-ELS perform slightly better than DIAG in terms of estimation precision. However on average DIAG only requires 5 JDTM iterations to converge against 46 and 140 for ALS-ELS and ALS, respectively. Hence Γ_{DIAG} is 10 to 100 times lower than Γ_{ALS} and $\Gamma_{ALS-ELS}$. Another interesting point is that the DIAG + ALS-ELS procedure limits the number of ALS iterations to 7-8 (the averaged number of iterations reported in the table for DIAG + ALS-ELS is the averaged number of ALS-ELS iterations used after an initialization with DIAG) and we can see from these results that this is enough to obtain a precision similar or better than that of ALS-ELS. Consequently, the numerical complexity of this approach is 3 to 10 times lower than than those of Γ_{ALS} and $\Gamma_{ALS-ELS}$. The last example is similar to the second one but tensor dimensions have been permuted so that only the DIAG unfolding matrix is different. Here DIAG results are degraded both in terms of precision and numerical complexity. We can conclude that if it is possible, it is better to not place the larger dimension of the tensor at the end.

Group of large third order tensors. We consider now third order tensors whose all the dimensions are equal to 50 or 100. As a consequence, the CPD rank is far lower than the tensor dimensions and all algorithms perform better and need fewer iterations. This explain that the gap between the different algorithms is narrowing. However we can still draw the same general conclusion: DIAG remains the cheapest solution and DIAG + ALS-ELS provides the same precision than ALS and ALS-ELS for a lower numerical complexity.

Group of fourth order tensors. We obtain the same kind of results that with the first group so that DIAG + ALS-ELS still appears to give the best compromise between precision and cost. One should note however that in the last case DIAG is by far the cheapest whereas its results regarding the NMSE are not as good. This is explained by the fact than the rank is greater than three of the tensor dimensions and slightly lower than the remaining one.

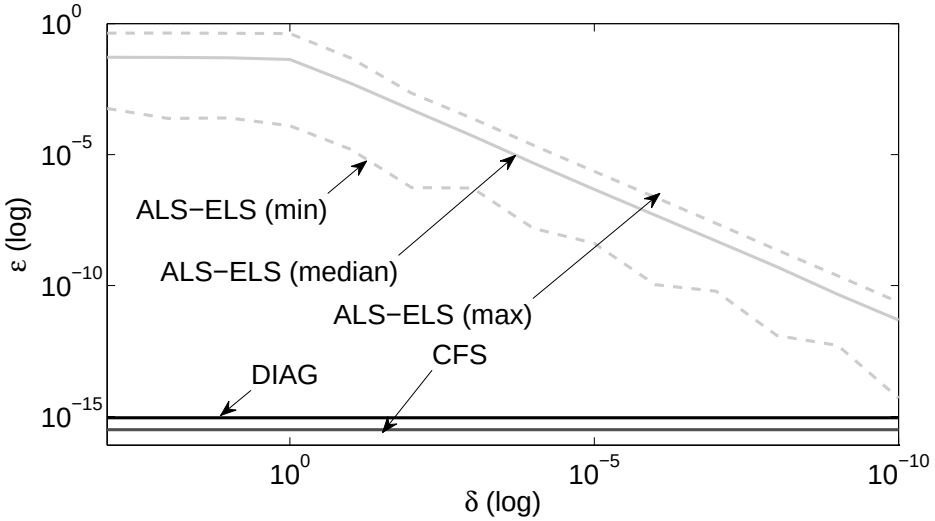


Figure 8: Decomposition of the Paatero tensor, evolution of the minimal median and maximal values of the ALS-ELS estimation error, according to a distance between the solution and the starting values and comparison with CFS and DIAG results.

Higher order tensors and tensors with correlated CPD factors. The first example of this group deals with fifth order tensors (for which our version of ELS is not working). For the second example we consider sixth order tensors. It is worth mentioning here that in both cases DIAG provides as accurate estimates as ALS and ALS-ELS do while its numerical complexity remains largely lower. Now looking at the last example with correlated factors, one can first note that ALS doesn't work whereas ALS-ELS is more accurate than DIAG. The price to paid is a very high number of iterations (444) and an increased computational cost (about 2×10^7 flops) against 5 iterations and about 3×10^5 flops for DIAG. In this case one should not that DIAG + ALS-ELS is significantly better than ALS-ELS in terms of estimation precision for a limited numerical complexity.

As a first conclusion DIAG appears as a good trade-off between estimation precision, speed and numerical complexity. Besides, the DIAG + ALS-ELS procedure provides a similar or better precision than that of ALS-ELS whereas its numerical complexity remains quite close to that of DIAG. Hence by combining both algorithms one can achieve the best precision, a good convergence speed and a reduced numerical complexity.

4.3. Results on the Paatero tensor

In [44], Paatero introduced a very simple 3-order tensor of size $2 \times 2 \times 2$ which has the following form:

$$\mathcal{T} = \left[\begin{array}{cc|cc} 0 & 1 & e & 0 \\ 1 & d & 0 & h \end{array}\right]. \tag{65}$$

Let's define its determinant Δ by:

$$\Delta = 4h + d^2e. \tag{66}$$

Then, it can be shown that the equation $\Delta = 0$ partitions the space into two subspaces, which hence have a non zero volume: The inequality $\Delta > 0$ defines the subspace of rank-2 tensors, whereas $\Delta < 0$ defines the subspace of rank-3 tensors. Finally, the closed set of tensors of rank 1 lies on the hypersurface $\Delta = 0$. [21]

Some of these tensors have the particularity of misleading any iterative algorithm, although the chosen starting value is close to the solution. As an example, Paatero notably consider in [44] to decompose the tensor defined by $(e, d, h) = (30, 0.26, 0.34)$ from the initial value $(e, d, h) = (30, 0.3, 0.12)$. This tensor belongs to the rank 2 subspace but it is close to the variety $\Delta = 0$. Its decomposition is given by the three following factor matrices:

$$\boldsymbol{A} = \begin{bmatrix} 1/x & -1/x \\ y_1 & y_2 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 1/x & -1/x \\ y_1 & y_2 \end{bmatrix} \quad \boldsymbol{C} = \begin{bmatrix} 1/x & -1/x \\ y_3 & y_4 \end{bmatrix}, \tag{67}$$

with: $x = (4h/e + d^2)^{\frac{1}{6}}$, $y_2 = (x^3 - d)/(2x)$, $y_1 = x^2 - a_2$, $y_4 = h/(y_2(y_1 + y_2))$ and $y_3 = y_2 y_4/y_1$.

Later in [21], authors confirmed that in this case, even the most efficient iterative algorithms such as ALS-ELS and Levenberg-Marquardt get stuck in a local minimum of the cost function, leading to a very bad estimation of the factor matrices. Actually, since the iterative algorithms works by successive optimization of rank-2 tensors, they cannot take the shorter paths to the solution which could cross the space of rank-3 tensors. Thereby, this is an other typical situation where direct algorithms can help. In order to see this we have reproduced the experiment here not only for the Paatero starting values but for different starting values around the solution. Hence we define a parameter δ such that the initial factor matrices of the ALS-ELS, $\boldsymbol{A}^{(0)}, \boldsymbol{B}^{(0)}$

$$\boldsymbol{A}^{(0)} = \boldsymbol{A} + \delta \boldsymbol{E}_A;\ \boldsymbol{B}^{(0)} = \boldsymbol{B} + \delta \boldsymbol{E}_B;\ \boldsymbol{C}^{(0)} = \boldsymbol{C} + \delta \boldsymbol{E}_C, \tag{68}$$

where $\boldsymbol{E}_A, \boldsymbol{E}_B$ and $\boldsymbol{E}_C$ are matrices of size 2×2 whose elements are randomly drawn according to a standard normal law. We now define ϵ as the mean estimation error upon the three estimated factor matrices, $\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{C}}$:

$$\epsilon = \frac{1}{3}\left(\frac{\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F}{\|\boldsymbol{A}\|_F} + \frac{\|\boldsymbol{B} - \widehat{\boldsymbol{B}}\|_F}{\|\boldsymbol{B}\|_F} + \frac{\|\boldsymbol{C} - \widehat{\boldsymbol{C}}\|_F}{\|\boldsymbol{C}\|_F} \right). \tag{69}$$

The ALS-ELS algorithm is run 500 times on the tensor $\boldsymbol{\mathcal{T}}$, with a new draw of the $\boldsymbol{E}_A, \boldsymbol{E}_B$ and $\boldsymbol{E}_C$ matrices at each time, and for different values of δ comprised between 1000 and 10^{-10}. We present on figure 8 the plots of the median, minimal and maximal values of $\epsilon_{ALS-ELS}$ according to the δ value. For comparison, both ϵ_{DIAG} and ϵ_{CFS} values are also reported on the figure. It can be seen that the iterative algorithm needs a very good initialisation in order to get an estimation precision close to the machine precision. Recall that we are looking for an exact decomposition since the considered tensors are noise free. Conversely, direct algorithms such as the closed-form solution or DIAG provide a perfect decomposition of $\boldsymbol{\mathcal{T}}$ and a thus an exact estimation of the factor matrices.

5. Application to fluorescence spectroscopy

A good application example of the CPD is found in fluorescence spectroscopy since after some numerical corrections measured data can be modelled by a CPD with physical meaning. Standard spectrofluorimeters allow to measure the intensity of the fluorescence signal emitted

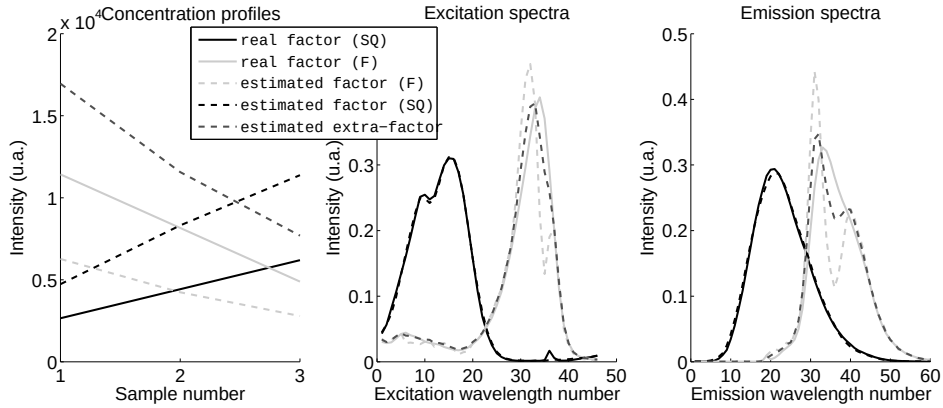


Figure 9: CPD factors of the fluorescence tensor using ALS.

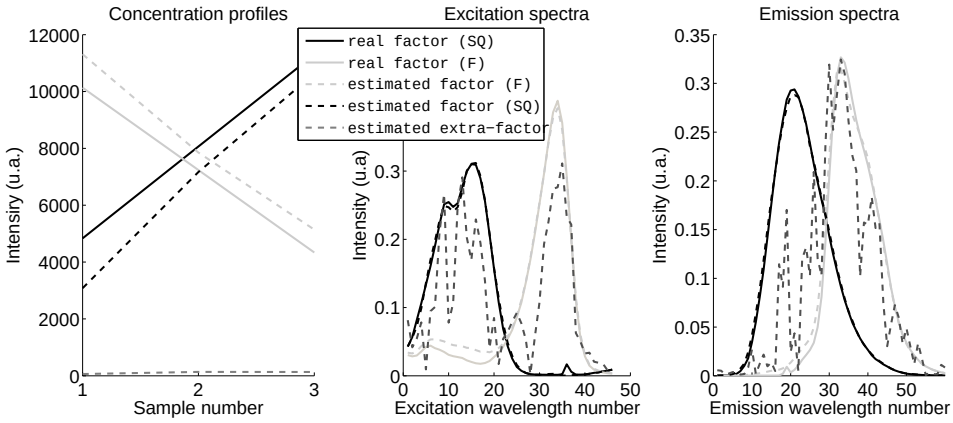


Figure 10: CPD factors of the fluorescence tensor using DIAG-JDTM.

by a diluted solution at wavelength λ_j by exciting the solution at wavelength λ_i. Hence, by scanning the excitation-emission couples (λ_i, λ_j) one obtains an $I \times J$ matrix of fluorescence which is called the Fluorescence Excitation-Emission Matrix (FEEM) of the solution. In many applications one have to measure a FEEM set corresponding to a set of K solutions and thus obtains a fluorescing data tensor $\boldsymbol{\mathcal{X}}$ of order 3 and size $I \times J \times K$ which contains the K FEEM. Solutions are often mixtures of a small number, R_m, of diluted fluorescing chemical species (fluorophores). Fluorophore concentrations vary from a solution to an other. Hence fluorophore r is characterized by its excitation spectrum, $e_r(\lambda_i)$, its fluorescence emission spectrum, $f_r(\lambda_j)$ and the variation of its concentration through the solution set (concentration profile), $c_r(k)$. In practice one wants to recover e_r, f_r and c_r $(r = 1 \cdots R_m)$ from the measured FEEMs. It can be shown that after removing scattering effects and correcting (or preventing) inner filter effects, the contribution of each fluorophore to the solution signal is linear in excitation, in emission and in concentration. In other words we have:

$$\mathcal{X}_{i,j,k} = \sum_{r=1}^{R_m} E_{i,r} F_{j,r} C_{k,r}, \tag{70}$$

where $E_{i,r} = e_r(\lambda_i)$, $F_{j,r} = f_r(\lambda_j)$ and $C_{k,r} = c_r(k)$, so that the CPD solves this inverse problem in a deterministic way. This is the reason why CPD has been largely applied to analyze FEEM sets since original works of Bro in this area [9, 10].

In most applications of fluorescence spectroscopy the number of fluorophores which defines the model rank of the decomposition is unknown and has to be estimated. However few methods exist and can give contradictory results and lead to over-factoring. A good example of this situation can be found in [45]. This would be acceptable if in each estimated factor matrix one obtain the R_m real factors aside with extra factors whose the contributions are almost null. Actually this is not the case with ALS which is very commonly used for analyzing this kind of data. Therefore this problem remains an important issue of FEEM analysis. In order to highlight the reliability of DIAG in this context we consider here a fluorescence tensor which contains the fluorescence intensity of 3 distinct mixtures of two fluorophores (fluorescein and quinine sulphate) measured at 46×71 excitation-emission wavelength couples. Hence the tensor size is $3 \times 46 \times 71$ and the model rank is 2. CPD of rank 3 were then used to decompose the tensor. ALS and DIAG results are reported on figure 9 and 10 respectively and compared to the actual factors after removing permutation and scaling indeterminacy. Excitation and Emission factors are normalized so that factor contributions are condensed in the concentration mode. Actual concentration profiles are perfectly known since these are laboratory mixtures and actual spectra were measured aside from pure solutions of fluorescein and quinine sulphate. This is a simple case for which both algorithms give perfect results when the good model rank ($R_m = 2$) is selected (data not shown). However ALS sensitivity to over-factoring effect in a concrete case clearly appears here. Indeed actual factors are not well estimated (notably the fluorescein spectra and the concentration profiles). Moreover contribution of the extra factor to the decomposition is significant. Recall that this factor has no physical meaning. On the opposite DIAG results are satisfying notably regarding the estimated spectra. One can verify that the contribution of the extra factor is almost null.

6. Conclusion

We have described in this paper a CPD algorithm that takes advantage of the link between CPD and Joint EVD in an original way. A JEVD algorithm has been conjointly proposed. Com-

putational complexities and extension to the complex field have been given for both algorithms. Numerical simulations point out the efficiency of the proposed JDTM algorithm to solve the JEVD problem. This algorithm usually offers more accurate results than its competitors especially in the most difficult cases involving big matrices and low SNR values. In terms of numerical complexity, the JDTM algorithm also provides good performances thanks to a remarkably low and stable number of iterations.

Classical iterative CPD algorithms such as ALS are usually efficient but suffer from convergence problem, notably in case of highly correlated factors, and are very sensitive to over-factoring. In addition they require a large number of iterations to reach the convergence. ELS allows us to reduce this number and deals with correlated factors in some situations but it remains useless in case of over-factoring. In addition, we have seen that there are some simple cases for which iterative approaches consistently fail for theoretical reasons.

In this context direct approaches such as the proposed DIAG algorithm have been designed to prevent such issues. First the DIAG algorithm involves a limited iterative procedure which requires very few iterations hence limiting global computational cost of the algorithm. Second it is insensitive to over-factoring thanks to the initial SVD which is independent of the chosen rank. These features have been verified in this paper by using many numerical simulations. Notably we have shown that DIAG was able to deal with highly correlated factors in all the modes or a large number of extra factor in case of over-factoring. Furthermore our results also demonstrates that DIAG is very efficient to decompose high order tensors. Finally it is a very fast algorithm with a lower computational complexity than ALS or ALS-ELS, notably in the case of small tensors or correlated factors.

As a counterpart, DIAG implies more restricted necessary conditions on the CPD rank than ALS. Therefore ALS-ELS is more accurate than DIAG when the rank is close to DIAG intrinsic limit. This is usually not the case in fluorescence spectroscopy applications for which at least one tensor dimension is largely greater than the model rank. In addition it has been shown that DIAG results can be improved by adding very few ALS iterations with a limited impact on the overall numerical complexity. Conversely, one should note that over-factoring is an important issue of FEEM analysis. This makes DIAG an attractive alternative to the classical ALS procedure for the CPD of fluorescence tensors, as it has been shown on a practical example.

Eventually, comparing to the CFS algorithm which is also a reference direct CPD approach, DIAG is a cheaper algorithm since it only involves one JEVD procedure and does not require to compare several estimates of the factor matrices. But its main advantage definitely comes from the necessary condition of CFS which is more restricted than DIAG's one. Hence there are many simple cases that CFS cannot handle. More generally CFS accuracy decreases as we get closer to its intrinsic limit. Otherwise CFS results are close to DIAG results.

Appendix A. Proof of proposition 1

Proof 2. *$C_{CFS} \Rightarrow C_{DIAG}$ is trivial. Indeed if $\exists\ (q_1, q_2) \in [1; Q]^2_{\mathbb{N}},\ q_1 \neq q_2$ such that $I(q_1) \geq R$ and $I(q_2) \geq R$ then taking any permutation f_I of the Q first natural number such that $f_I(1) = q_1$ and $f_I(2) = q_2$, $P = 1$ and $q_s = Q$ ensures C_{DIAG}.*

Let now suppose that C_{DIAG} is true and that C_{ALS} is false, i.e.:

$$\exists\, P \in [2; Q-1]_{\mathbb{N}},\ \exists\ f_I,\ \exists\, q_s > P \text{ and } q \leq Q \text{ such that:} \quad 1) \quad \prod_{i=1}^{P} I(f_I(i)) \geq R, \tag{A.1}$$

$$2) \quad \prod_{\substack{i=P+1 \\ i \neq q_s}}^{Q} I(f_I(i)) \geq R, \tag{A.2}$$

$$3) \quad \prod_{i=1}^{Q} I(i) < RI(q). \tag{A.3}$$

Since we have $1 \leq q \leq Q$ thus $q \in \{f_I(1), \cdots, f_I(P)\} \cup \{f_I(P+1), \cdots, f_I(Q)\}$.

- *We first assume that* $q \in \{f_I(1), \cdots, f_I(P)\}$. 2) *and* 3) *give:*

$$\frac{1}{I(f_I(q_s))} \prod_{i=P+1}^{Q} I(f_I(i)) > \frac{1}{I(q)} \prod_{i=1}^{Q} I(i), \tag{A.4}$$

$$\frac{1}{I(f_I(q_s))} \prod_{i=P+1}^{Q} I(f_I(i)) > \frac{1}{I(q)} \prod_{i=1}^{Q} I(f_I(i)), \tag{A.5}$$

$$I(q) > I(f_I(q_s)) \prod_{i=1}^{P} I(f_I(i)). \tag{A.6}$$

Since $q \in \{f_I(1), \cdots, f_I(P)\}$,

$$\prod_{i=1}^{P} I(f_I(i)) = I(q) \prod_{\substack{i=1 \\ i \neq f_I^{-1}(q)}}^{P} I(f_I(i)) \tag{A.7}$$

thereby,

$$1 > I(f_I(q_s)) \prod_{\substack{i=1 \\ i \neq f_I^{-1}(q)}}^{P} I(f_I(i)) \tag{A.8}$$

which is absurd.

- *Now we assume that* $q \in \{f_I(P+1), \cdots, f_I(Q)\}$. *Thereby,*

$$I(q) < \prod_{i=P+1}^{Q} I(f_I(i)), \tag{A.9}$$

$$I(q) \prod_{i=1}^{P} I(f_I(i)) < \prod_{i=1}^{Q} I(f_I(i)), \tag{A.10}$$

while 1) *and* 3) *give:*

$$I(q) \prod_{i=1}^{P} I(f_I(i)) > \prod_{i=1}^{Q} I(i), \tag{A.11}$$

$$I(q) \prod_{i=1}^{P} I(f_I(i)) > \prod_{i=1}^{Q} I(f_I(i)), \tag{A.12}$$

which is contradictory to (A.10).

Therefore if C_{DIAG} is verified then C_{ALS} is verified.

References

[1] Carroll, J.D., Chang, J.J.: Analysis of Individual Differences in Multidimensional Scaling via N-Way Generalization of Eckart-Young Decomposition, Psychometrika, 35 (3) pp. 283-319 (1970)

[2] Becker, A., Comon, P., Albera, L., Haardt, M., Merlet, I.: Multi-way Space-Time-Wave-Vector analysis for EEG source separation. Signal Processing, to appear in

[3] Deburchgraeve W., Cherian P.J., De Vos M., Swarte R.M., Blok J.H., Visser G.H., Govaert P., Van Huffel S.: Neonatal seizure localization using PARAFAC decomposition, Clinical Neurophysiology, 120, pp. 1787-1796 (2009)

[4] Vanderperren K., De Vos M., Mijovic B., Ramautar J.R., Novitskiy N., Vanrumste B., Stiers P., Van den Bergh B.R.H., Wagemans J., Lagae L., Sunaert S., Van Huffel S.: PARAFAC on ERP data from a visual detection task during simultaneous fMRI acquisition, in Proc. of the International Biosignal Processing Conference., Berlin, Germany, Jul. 2010, 103, pp. 1-4 (2010)

[5] Karfoul, A. , Albera, L., De Lathauwer, L.: Iterative methods for the canonical decomposition of multi-way arrays: Application to blind underdetermined mixture identification. Signal Processing, 91 (8), pp. 1789-1802 (2011)

[6] Sidiropoulos, N.D., Giannakis, G.B., Bro, R.: Blind PARAFAC receivers for DS-CDMA systems, IEEE Transactions On Signal Processing, 48 (8), pp. 810-823 (2000)

[7] Nion, D., Sidiropoulos, N.D., Tensor Algebra and Multi-dimensional Harmonic Retrieval in Signal Processing for MIMO Radar, IEEE Trans. on Signal Processing, 58 (11), pp. 5693-5705 (2010)

[8] Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory, 315, Springer (1997)

[9] Bro, R.: PARAFAC, Tutorial and Applications, Chemom. Intel. Lab. Syst., 38, pp. 149-171 (1997)

[10] Stedmon, C.A., Markager, S., Bro, R.: Tracing dissolved organic matter in aquatic environments using a new approach to fluorescence spectroscopy, Marine Chemistry, 82 (3-4), pp. 239-254 (2003)

[11] Harshman, R.A.: Determination and proof of minimum uniqueness conditions for PARAFAC-1," UCLA Working Papers in Phonetics, 22, pp. 111-117 (1972)

[12] Kruskal, J.B.: Three-Way Arrays: Rank and Uniqueness of Trilinear Decompositions, Linear Algebra and Applications, 18, pp. 95-138 (1977)

[13] Ten Berge, J.M.F., Sidiropoulos, N.D.: On uniqueness in CANDECOMP/PARAFAC, Psychometrika, 67, pp. 399-409 (2002)

[14] Jiang, T., Sidiropoulos, N.D.: Kruskal's permutation lemma and the identification of CANDECOMP/PARAFAC and bilinear models with constant modulus constraints, Trans. on Sig. Proc., 52 (9), pp. 2625-2636 (2004)

[15] De Lathauwer, L.: A Link between Canonical Decomposition in Multilinear Algebra and Simultaneous Matrix Diagonalization, Journal on Matrix Analysis, 28 (3), pp. 642-666 (2006)

[16] Stegeman, A., Sidiropoulos, N.D.: On Kruskal's uniqueness condition for the Candecomp/Parafac decomposition, Linear Algebra and its Applications, 420, pp. 540-552 (2007)

[17] Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products, J. Math. Phys, 6 (1), pp. 164-189 (1927)

[18] Harshman, R.A.: Foundations of the Parafac procedure: Models and conditions for an explanatory multimodal factor analysis, UCLA Working Papers in Phonetics, 16, pp. 1-84 (1970)

[19] Tendeiro, J., Bennani Dosse, M., Ten Berge, J.M.F., First and second-order derivatives for CP and INDSCAL, Chemometrics and Intelligent Laboratory Systems, 102 (1), pp. 27-36 (2011)

[20] Tomasi, G., Bro, R.: A comparison of algorithms for fitting the PARAFAC model, Comp. Stat. Data Anal., 50, pp. 1700-1734 (2006)

[21] Comon, P., Luciani, X., De Almeida, A.L.F.: Tensor Decompositions, Alternating Least Squares and other Tales, Journal of Chemometrics, 23 (9), pp. 393-405 (2009)

[22] Acar E., Dunlavy D.M., Kolda, T.G.: Scalable Tensor Factorizations with Missing Data, In SDM10: Proceedings of the 2010 SIAM International Conference on Data Mining, pp. 701-712 (2010)

[23] Kindermann, S., Navasca, C.: New algorithms for tensor decomposition based on a reduced functional, Numer. Linear Algebra Appl. (2013)

[24] Rajih, M., Comon, P., Harshman, R.: Enhanced Line Search : A Novel Method to Accelerate PARAFAC, SIAM Journal on Matrix Analysis Appl., 30 (3), pp. 1148-1171 (2008)

[25] Faber, N.M., Buydens, L.M.C., Kateman, G.: Generalized rank annihilation method: II. Bias and variance in the estimated eigenvalues, J. Chemom., 8, pp.181-203 (1994)

[26] Faber N.M., Bro R., Hopke P.K: Recent developments in CANDECOMP/PARAFAC algorithms: a critical review, Chemom. Intel. Lab. Syst., 65 (1), pp. 119-137 (2003)
[27] Roemer, F., Haardt, M.: A closed-form solution for Parallel Factor (PARAFAC) Analysis, In IEEE ICASSP'2008, pp. 2365-2368 (2008)
[28] Roemer, F., Haardt, M.: A closed-form solution for multilinear PARAFAC decompositions, In IEEE SAM 2008, pp. 487-491 (2008)
[29] De Lathauwer, L., De Moor, B., Vandewalle, J.: Computation of the Canonical Decomposition by Means of a Simultaneous Generalized Schur Decomposition, SIAM Journal on Matrix Analysis and Applications, 26, pp. 295-327 (2001)
[30] Albera, L., Ferréol, A., Comon, P., Chevalier, P.: Blind Identification of Overcomplete MixturEs of sources (BIOME), Linear Algebra and its Applications, 391, pp. 3-11 (2004)
[31] Luciani, X., Albera, A.: Joint Eigenvalue Decomposition using Polar Matrix Factorization, In LVA/ICA 2010, pp. 612-619 (2010)
[32] Luciani, X., Albera, A.: Semi-algebraic canonical decomposition of multi-way arrays and joint eigenvalue decomposition, In IEEE ICASSP'2011, pp. 4104-4107 (2011)
[33] Karfoul, A., Albera, L., Birot, G.: Blind underdetermined mixture identification by joint canonical decomposition of HO cumulants, IEEE Trans. Signal Proc., 58 (2), pp. 638-649 (2010)
[34] Ruhe, A.: On the quadratic convergence of a generalization of the Jacobi method to arbitrary matrices, BIT Numerical Mathematics, 8, pp. 210-231 (1968)
[35] Haardt, M., Nossek, J. A.: Simultaneous Schur decomposition of several nonsymmetric matrices to achieve automatic pairing in multidimensional harmonic retrieveal problems, IEEE Trans. Signal Proc., 46, pp. 161-169 (1998)
[36] Strobach, P.: Bi-iteration multiple invariance subspace tracking and adaptive ESPRIT, IEEE Trans. Signal Proc., 48, pp. 442-456 (2000)
[37] Fu, T., Gao, X.: Simultaneous Diagonalization with Similarity Transformation for Non-defective Matrices. In IEEE ICASSP'2006, pp. 1137-1140 (2006)
[38] Goldstine, H.H.: Horwitz, L.P.: A procedure for the diagonalization of normal matrices, J. ACM, 6 (2), pp. 176-195 (1959)
[39] Eberlein, P.J.: A Jacobi-like method for the automatic computation of eigenvalues and eigenvectors of an arbitrary matrix, Journal of the Society for Industrial and Applied Mathematics, 10 (1), pp. 74-88 (1962)
[40] Souloumiac, A.: Nonorthogonal joint Diagonalization by Combining Givens and Hyperbolic Rotations, IEEE Trans. Signal Proc., 57 (6), pp. 2222-2231 (2009)
[41] Iferroudjene, R., Abed Meraim, K., Belouchrani, A.: A New Jacobi-like Method for Joint Diagonalization of Arbitrary non-defective Matrices, Applied Mathematics and Computation 211, pp. 363-373 (2009)
[42] Cardoso, J.-F., Souloumiac, A.: Jacobi Angles for Simultaneous Diagonalization, SIAM Journal on Matrix Analysis and Applications, 17 (1), pp. 161-164 (1996)
[43] De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition, SIAM Journal on Matrix Analysis and Applications, 21 (4), pp. 1253-1278, (2000)
[44] Paatero, P.: Construction and analysis of degenerate Parafac models, Journal of Chemometrics, 14, pp. 285-299 (2000)
[45] Luciani, X., Mounier, S., Paraquetti, H.H.M., Redon, R., Lucas, Y., Bois, A., Lacerda, L.D., Raynaud, M., Ripert, M.:Tracing of dissolved organic matter from the SEPETIBA Bay (Brazil) by PARAFAC analysis of total luminescence matrices, Marine Environmental Research, 65 (2), pp. 148-157 (2008)